Another Look at Tightness [⋆]

Sanjit Chatterjee[1], Alfred Menezes[2], and Palash Sarkar[3]

[1] Department of Computer Science and Automation, Indian Institute of Science
sanjit@csa.iisc.ernet.in
[2] Department of Combinatorics & Optimization, University of Waterloo
ajmeneze@uwaterloo.ca
[3] Applied Statistics Unit, Indian Statistical Institute
palash@isical.ac.in

Abstract. We examine a natural, but non-tight, reductionist security proof for deterministic message authentication code (MAC) schemes in the multi-user setting. If security parameters for the MAC scheme are selected without accounting for the non-tightness in the reduction, then the MAC scheme is shown to provide a level of security that is less than desirable in the multi-user setting. We find similar deficiencies in the security assurances provided by non-tight proofs when we analyze some protocols in the literature including ones for network authentication and aggregate MACs. Our observations call into question the practical value of non-tight reductionist security proofs. We also exhibit attacks on authenticated encryption schemes, disk encryption schemes, and stream ciphers in the multi-user setting.

1 Introduction

A reductionist security proof for a cryptographic protocol $\mathcal{P}$ with respect to a problem $\mathcal{S}$ is an algorithm $\mathcal{R}$ for solving $\mathcal{S}$, where $\mathcal{R}$ has access to a hypothetical subroutine $\mathcal{A}$ (called an oracle) that achieves the adversarial goal specified by the security definition for $\mathcal{P}$. Suppose that $\mathcal{A}$ takes time at most T and is successful with probability at least ϵ, where T and ϵ are functions of the security parameter. Suppose further that $\mathcal{R}$ solves $\mathcal{S}$ in time T' with probability at least ϵ'; again, T' and ϵ' are functions of the security parameter. Then the reductionist security proof $\mathcal{R}$ is said to be *tight* if $T' \approx T$ and $\epsilon' \approx \epsilon$. Roughly speaking, it is *non-tight* if $T' \gg T$ or if $\epsilon' \ll \epsilon$, in which case the *tightness gap* can be informally defined to be $(T'\epsilon)/(T\epsilon')$.

A tight proof for $\mathcal{P}$ with respect to $\mathcal{S}$ is desirable because one can then deploy $\mathcal{P}$ and be assured that breaking $\mathcal{P}$ (within the confines of the adversarial model specified by the security definition for $\mathcal{P}$) is at least as hard as solving $\mathcal{S}$. On the other hand, a non-tight proof for $\mathcal{P}$ with respect to $\mathcal{S}$ provides only the weaker assurance that breaking $\mathcal{P}$ requires at least as much work as a certain fraction

[⋆] August 14, 2011; updated on April 1, 2013. This paper is an updated and corrected version of the paper that appeared in the Proceedings of SAC 2011 (Lecture Notes in Computer Science, volume 7118, pages 293-319).

of the work believed to be necessary for solving $\mathcal{S}$. In that case, the desired security assurance for $\mathcal{P}$ can be attained by using larger parameters — but at the expense of slower performance.

BBS generator. As an example, consider the Blum-Blum-Shub (BBS) pseudorandom bit generator G [14]. For an n-bit integer N that is the product of two primes each of which is congruent to 3 modulo 4, the BBS generator takes a random integer $x \bmod N$ as the seed and produces $M = jk$ bits as follows: Let $x_0 = x$, and for $i = 1, \ldots, k$ let

$$x_i = \min\{x_{i-1}^2 \bmod N,\ N - (x_{i-1}^2 \bmod N)\}.$$

Then the output of G consists of the j least significant bits of x_i, $i = 1, \ldots, k$.

In [1] it was proven that $j = O(\log n)$ bits can be securely extracted in each iteration, under the assumption that factoring is intractable. More precisely, if one assumes that no algorithm can factor N in expected time less than $L(n)$, then the security proof in [1] (see [71]) shows that the BBS generator is (T, ϵ) secure if

$$T \leq \frac{L(n)(\epsilon/M)^8}{2^{4j+27}n^3}. \tag{1}$$

Here, (T, ϵ)-security means that there is no algorithm with running time bounded by T which can distinguish between the outputs of G and a purely random bit generator with advantage greater than ϵ.

The aforementioned security proof is an example of a polynomial-time reduction since, for $M = O(n^c)$ (where c is a constant), $j = O(\log n)$, and constant ϵ, the right-hand side of (1) is of the form $L(n)/f(n)$ where f is a polynomial in the security parameter n. Such polynomial-time security proofs provide security assurances in an asymptotic sense, i.e., as the security parameter n tends to infinity. However, for a fixed security parameter that might be used in practice, the proof might provide little or no security assurance. For example, suppose that one were to follow the recommendations in [29] and [75] and implement the BBS generator with $n = 768$ and $j = 9$. Then, as observed in [49], by using the number field sieve to estimate $L(n)$ and taking $M = 10^7$ and $\epsilon = 0.01$, one sees that the inequality (1) provides security assurances only against an adversary whose time is bounded by 2^{-264}. Thus, the security proof is completely meaningless for these parameters.

Does tightness matter? As discussed in [49], a non-tight reductionist security proof for a protocol $\mathcal{P}$ with respect to a problem $\mathcal{S}$ can be interpreted in several ways. An optimistic interpretation is that it is reasonable to expect that a tighter reduction will be found in the future, or perhaps that $\mathcal{P}$ is secure *in practice* in the sense that there is no attack on $\mathcal{P}$ that is faster than the best attack on $\mathcal{S}$ even though a tight reduction from $\mathcal{S}$ to $\mathcal{P}$ might not exist. However, strictly speaking, if one implements $\mathcal{P}$ using a security parameter for which the problem $\mathcal{S}$ is expected to take time T' to solve, then the security proof does not rule out the possibility that an attack on $\mathcal{P}$ which takes time considerably less than T' will be discovered in the future.

Researchers who work in theoretical cryptography are generally satisfied with polynomial-time reductions, although some of them caution about the validity of non-tight proofs in practice. For instance, Luby [55] writes "when we describe a reduction of one primitive to another we are careful to quantify how much of the security of the first instance is transferred to the second." Goldreich [36] cautions that a (non-tight) asymptotic proof offers only the "plausibility" of the protocol's security. On the other hand, Damgård [25] asserts that a non-tight polynomial-time reduction is useful because it rules out all polynomial-time attacks. However, such an assurance is not very comforting since proofs are meant to guarantee resistance to *all* attacks, and moreover there are many examples of practical cryptographic schemes that have succumbed to attacks that are deemed to be effective in practice even though in asymptotic terms they require super-polynomial time.

Considerable effort has been expended on devising tighter security proofs for existing protocols, and on designing new protocols with tighter security proofs. For example, the first security proof [5] for the traditional hash-then-sign RSA signature scheme (called RSA-FDH) was highly non-tight. Subsequently, Coron [23] found an alternate proof that is significantly tighter (although still considered non-tight). Coron [24] proved that no tighter reduction exists; however, Kakvi and Kiltz [42] have recently shown that Coron's argument for the non-existence of a tighter reduction for RSA-FDH is flawed when the encryption exponent e has bitlength less than a quarter that of the RSA modulus N (see also [43]). Meanwhile, Katz and Wang [45] showed that a small modification of RSA-FDH yields a signature scheme that has a tight security proof, arguably increasing confidence in RSA-FDH itself. Nonetheless, another variant of RSA-FDH, called RSA-PSS, is commonly recommended in practice because it has a tight security proof [5]. As another example, Gentry and Halevi [35] designed a hierarchical identity-based encryption (HIBE) scheme that has a security proof whose tightness gap depends only linearly on the number of levels, in contrast to all previous HIBE schemes whose tightness gaps depend exponentially on the number of levels. Finally, we mention Bernstein's [7] tight proof in the random oracle model for the Rabin-Williams signature scheme, and Schäge's [68] tight proofs without the random oracle assumption for the Cramer-Shoup and Camenisch-Lysyanskaya signature schemes.

Despite ongoing efforts by some to tighten security proofs of existing protocols and to develop new protocols with tighter proofs, it is fair to say that, for the most part, the tightness gaps in security proofs are not viewed as a major concern in practice. Researchers who design protocols with non-tight proofs typically give arguments in favour of their protocol's efficiency by using parameters that would make sense if the proof had been tight. For example, the Schnorr signature scheme [69] is widely regarded as being secure, although its known security proofs are highly non-tight [61]. In fact, there are arguments which suggest that a tighter proof is not even possible [60]. Nevertheless, the Schnorr signature scheme is widely used in the cryptographic literature without any suggestion to use larger key sizes to account for the tightness gap in the proof.

Other examples of well-known protocols with highly non-tight proofs include the Boneh-Franklin (BF) [16, 34], Sakai-Kasahara (SK) [22] and Boneh-Boyen (BB1) [15] identity-based encryption schemes, the Lu et al. aggregate signature scheme [54], and the HMQV key agreement protocol [51]. In [18], Boyen compares the tightness of the reductions for BB1, BF and SK. The reduction for BB1 is significantly tighter than the reduction for BF, which in turn is significantly tighter than that for SK. However, all three reductions are in fact highly non-tight, the tightness gap being (at least) linear, quadratic and cubic in the number of random oracle queries made by the adversary for BB1, BF and SK, respectively. Although all these proofs have large tightness gaps, Boyen recommends that SK should "generally be avoided as a rule of thumb", BF is "safe to use", and BB1 "appears to be the smartest choice" in part due to the "fairly efficient security reduction" of the latter. Despite the importance Boyen attaches to tightness as a reason for avoiding SK, a recent IETF standard co-authored by Boyen that describes BB1 and BF [19] does not recommend larger security parameters to account for tightness gaps in their security proofs.

Our work. In §2, we examine a natural, but non-tight, reductionist security proof for MAC schemes in the multi-user setting. If parameters are selected without accounting for the tightness gap in the reduction, then the MAC scheme is shown to provide a level of security that is less than what one would desire in the multi-user setting. In particular, the attacks we describe are effective on HMAC as standardized in [33, 26] and CMAC as standardized in [28, 72]. In §3, we show that this deficiency in the security assurances provided by the non-tight proof appears in a network authentication protocol [20], and in §4 we obtain analogous results for aggregate MACs and aggregate designated verifier signatures. In §5, we exhibit attacks on some authenticated encryption schemes, disk encryption schemes, and stream ciphers in the multi-user setting. We draw our conclusions in §6.

2 MACs in the multi-user setting

Cryptographic protocols that provide basic confidentiality and authentication security services are typically examined in the *single-user setting*, where there is only one legitimate user (or a pair of legitimate users) and an adversary. However, these protocols are generally deployed in the *multi-user setting*, where there may be additional threats. Key establishment protocols were first analyzed in the multi-user setting in [4, 13]. This was followed by a study of multi-user public-key encryption [2] and signatures [57]. In this section, we consider the security of MAC schemes in the multi-user setting.

2.1 Security definition

A *MAC scheme* consists of a family of functions $\{H_k\}_{k\in\mathcal{K}}$, where $\mathcal{K} = \{0,1\}^r$ is the key space and $H_k : \mathcal{D} \to \{0,1\}^t$ for each $k \in \mathcal{K}$. Here, $\mathcal{D}$ is the set of all (non-empty) binary strings of some maximum length L. A pair of users A and

B select a secret key $k \in \mathcal{K}$. To authenticate a message $m \in \mathcal{D}$, user A computes the tag $\tau = H_k(m)$ and sends (m, τ). The receiver B verifies that $\tau = H_k(m)$.

The traditional definition of MAC security (in the single-user setting) is the following. An adversary $\mathcal{B}$ has complete knowledge of the MAC scheme, i.e., it can select arbitrary $k \in \mathcal{K}$ and $m \in \mathcal{D}$ and compute $H_k(m)$. Now, a key k' is selected independently and uniformly at random from $\mathcal{K}$ and kept secret from $\mathcal{B}$. The adversary $\mathcal{B}$ has access to a MAC oracle indexed by k' in the following way: for any $m \in \mathcal{D}$ of $\mathcal{B}$'s choosing, $\mathcal{B}$ is given $H_{k'}(m)$. $\mathcal{B}$'s goal is to produce a *forgery*, i.e., a pair (m, τ) such that $m \in \mathcal{D}$ was not queried to the MAC oracle and $H_{k'}(m) = \tau$. We will henceforth denote $\mathcal{B}$'s task by *MAC1* (breaking a MAC scheme in the single-user setting). An adversary $\mathcal{B}$ is said to (T, ϵ)-break MAC1 if its running time is bounded by T and it produces a forgery with probability at least ϵ; the probability is assessed over the choice of k' and $\mathcal{B}$'s coin tosses. MAC1 is said to be (T, ϵ)-secure if there does not exist an adversary $\mathcal{B}$ that (T, ϵ)-breaks it.

Our definition of MAC security in the multi-user setting is the following. An adversary $\mathcal{A}$ has complete knowledge of the MAC scheme. First, n keys $k_1, k_2, \ldots, k_n$ corresponding to users[1] $1, 2, \ldots, n$ are chosen independently and uniformly at random from $\mathcal{K}$ and kept secret from $\mathcal{A}$; n is an upper bound on the total number of users in the system. The adversary $\mathcal{A}$ has access to MAC oracles indexed by $k_1, \ldots, k_n$ in the following way: for any (i, m) of $\mathcal{A}$'s choosing, where $i \in [1, n]$ and $m \in \mathcal{D}$, $\mathcal{A}$ is given $H_{k_i}(m)$. Furthermore, $\mathcal{A}$ is allowed to *corrupt* any oracle (or user); i.e., for any $i \in [1, n]$ of its choosing, $\mathcal{A}$ is given k_i. The adversary's goal is to produce a *forgery*, i.e., find a triple (i, m, τ) such that:

(i) $i \in [1, n]$ and $m \in \mathcal{D}$;
(ii) the adversary did not corrupt oracle i;
(iii) the adversary did not query H_{k_i} with m; and
(iv) $H_{k_i}(m) = \tau$.

Henceforth, $\mathcal{A}$'s task will be denoted by *MAC** (breaking a MAC scheme in the multi-user setting)[2]. $\mathcal{A}$ is said to (T, ϵ)-break MAC* if its running time is bounded by T and it produces a forgery with probability at least ϵ; the probability is assessed over the choices of $k_1, \ldots, k_n$ and $\mathcal{A}$'s coin tosses. MAC* is said to be (T, ϵ)-secure if there does not exist an adversary $\mathcal{A}$ that (T, ϵ)-breaks it.

2.2 Reductionist security proof

We present a natural reductionist security proof that a MAC scheme is secure in the multi-user setting, provided that it is secure in the single-user setting.

Suppose, by way of contradiction, that $\mathcal{A}$ is an adversary that (T, ϵ)-breaks MAC*. Suppose we are given access to a MAC oracle for H_k, where $k \in_R \mathcal{K}$; call the oracle MAC$_k$. We show how $\mathcal{A}$ can be used to design an adversary $\mathcal{B}$ that produces a forgery with respect to MAC$_k$.

[1] More precisely, a 'user' is a pair of entities who share a symmetric key.

[2] The MAC* problem without the corrupt capability was first formulated in [13].

$\mathcal{B}$ begins by selecting an index $j \in_R [1, n]$, guessing that if $\mathcal{A}$ succeeds then its forgery will be with respect to user j. For each $i \in [1, n]$ with $i \neq j$, $\mathcal{B}$ selects $k_i \in_R \mathcal{K}$ as i's secret key. User j's secret key is assigned to be k (which is unknown to $\mathcal{B}$). $\mathcal{B}$ now runs $\mathcal{A}$, answering $\mathcal{A}$'s MAC and corrupt queries to users $i \neq j$ using knowledge of k_i, and using the given oracle MAC_k to answer $\mathcal{A}$'s MAC queries to user j. If $\mathcal{A}$ corrupts user j, then $\mathcal{B}$ aborts with failure. If $\mathcal{A}$ outputs a forgery (j, m, τ), then $\mathcal{B}$ outputs (m, τ) as its forgery with respect to MAC_k; otherwise, $\mathcal{B}$'s experiment has failed.

Now, $\mathcal{A}$'s operation is independent of $\mathcal{B}$'s guess j, unless $\mathcal{A}$ corrupts user j in which case $\mathcal{B}$ is certain to fail. Hence, the probability that $\mathcal{A}$ succeeds and $\mathcal{B}$'s guess is correct is at least ϵ/n and so $\mathcal{B}$ $(T, \epsilon/n)$-breaks MAC1. We conclude that if a MAC scheme is (T, ϵ)-secure in the single-user setting, then it is $(T, n\epsilon)$-secure in the multi-user setting.

Remark 1. (*tightness gap in the security proof for MAC**) The security reduction is non-tight, having a tightness gap of n, the number of users of the MAC scheme. In §2.3, we present a generic attack on an ideal MAC scheme in the multi-user setting that, under the assumption that keys and tags are of the same bitlength r, produces a MAC forgery within time $2^r/n$. The attack is faster than the best possible attack — exhaustive key search with running time 2^r — on an ideal MAC scheme in the single-user setting. Note that the attack does not contradict the security proof for MAC* because of the tightness gap of n. The attack suggests that a reduction for MAC* that is tighter than the one given above does not exist.

Remark 2. (*tightness gap in the security proof for RSA-FDH*) The security proof for MAC* given above is somewhat similar to the Bellare-Rogaway security proof for RSA-FDH in the random oracle model [5]. Recall that the RSA-FDH signature on a message m is $s = H(m)^d \bmod N$, where (N, e) is an RSA public key and d is the corresponding private key, and where $H : \{0,1\}^* \to [0, N-1]$ is a hash function modeled as a public random oracle. In the Bellare-Rogaway proof, the simulator uses a signature forger $\mathcal{F}$ to solve a given instance (N, e, y) of the RSA problem, i.e., find $x \in [0, N-1]$ satisfying $y \equiv x^e \pmod{N}$. The forger $\mathcal{F}$ is executed with (N, e) as public key, and the simulator has to faithfully answer $\mathcal{F}$'s signature queries and queries to H. Assuming that $\mathcal{F}$ makes at most q H-queries, the simulator selects $j \in_R [1, q]$ and answers the jth H-query m with $H(m) = y$, hoping that $\mathcal{F}$ eventually produces a forger on m — since the signature on m must be x, the simulator thereby obtains the solution to its instance of the RSA problem. If $\mathcal{F}$ forges a signature on any of the other $q - 1$ messages it presented to H, then the simulator fails. Consequently, the security reduction has a tightness gap of q.

Coron [23] gave an alternate security proof for RSA-FDH for which the tightness gap is q_S, the number of signature queries $\mathcal{F}$ is permitted to make. Since q_S can be expected to be significantly smaller than the number of hash queries a real-world forger can make, Coron's proof is significantly tighter. However, it is still non-tight. For the case $e < N^{1/4}$, it is not known whether a tighter proof with respect to the RSA problem exists [42, 43].

Unlike the non-tight proof for multi-user MAC schemes, the tightness gap in the RSA-FDH proof does not seem to be a concern because no one expects there to be a method for breaking RSA-FDH that is faster than solving the RSA problem (for which the fastest method known is to factor N). Indeed, it is shown in [48] that RSA-FDH is *tightly* equivalent to an interactive version of the RSA problem, called RSA1. Although Coron's separation result implies that the RSA1 problem cannot in general be proven to be tightly equivalent to the RSA problem, reasonable heuristic arguments suggest that the RSA1 and RSA problems are indeed equivalent in practice. We also note that Kakvi and Kiltz [42] have provided a tight security proof with respect the the "Φ-hiding assumption" in the case where e is prime and $e < N^{1/4}$.

Remark 3. (*tightness gaps in security proofs for Diffie-Hellman key agreement protocols*) Numerous Diffie-Hellman key agreement protocols have been proposed in the literature, and many of them have been proven secure in the Canetti-Krawczyk (CK) [20] model (and its variants). In the CK model, there can be many users, any two of which can engage in several sessions of the key agreement protocol; suppose that there are at most n sessions in total. The security proofs are with respect to the computational Diffie-Hellman problem (CDH) or a variant of it: given g, g^x and g^y, where g is a generator of a cyclic group, compute g^{xy}. Typically, one part of the proof involves the simulator selecting a session j at random and then embedding g^x and g^y as the (ephemeral or static) public keys of each of the two communicating parties for that session. If the adversary of the key agreement protocol succeeds in compromising the security of the jth session, then the simulator is able to compute g^{xy}; otherwise the simulator fails. Consequently, the security reduction has a tightness gap of at least n.

To the best of our knowledge, all published security proofs for Diffie-Hellman protocols in the CK model (e.g., see [51, 53, 58, 73]) have tightness gaps of at least n. However, no one has insisted that implementations of these protocols use larger security parameters in order to account for the possible existence of an attack that is better than the fastest known attack on the underlying Diffie-Hellman problem.

2.3 An attack on MAC*

Select an arbitrary message m and obtain the tags $H_{k_i}(m)$ for $i = 1, 2, \ldots, n$. Next, select an arbitrary subset $\mathcal{W}$ of keys with $|\mathcal{W}| = w$. For each $\ell \in \mathcal{W}$, compute $H_\ell(m)$. If $H_\ell(m) = H_{k_i}(m)$ for some i (this event is called a *collision*), then conclude that $\ell = k_i$ and use ℓ to forge a message-tag pair for user i.

The expected running time of the attack is w, and there are n MAC queries. The attack is deemed successful if $\ell = k_i$ the first time a collision $H_\ell(m) = H_{k_i}(m)$ is detected. In §2.4, the attack's success probability is analyzed in the ideal MAC model. Recall that r is the key length and t is the tag length. Suppose that $n^2 \ll 2^{r+1}$, and suppose that $nw = c2^{\min\{r,t\}}$ for some constant c. One consequence of the analysis is that if $r = t$, then the success probability is approximately $\frac{1}{2}$. If $t \gg r$ then the success probability is essentially 1, whereas if $r \gg t$ then the attack is virtually certain to fail[3]. These conclusions are not surprising, as the following informal argument shows. A collision can occur due to either a *key collision* (i.e., $k_i = \ell$) or a *tag collision* (i.e., $H_{k_i}(m) = H_\ell(m)$ but $k_i \neq \ell$). Given that a collision has occurred, if keys and tags are of the same size, then the probability that it is due to a key collision is about $\frac{1}{2}$; if keys are much longer than tags, the collision is most likely due to a tag collision; and if tags are much longer than the keys, then the collision is most likely due to a key collision.

In the remainder of the paper, the attack will be referred to as *Attack 1.*

Remark 4. (*a second attack on MAC**) Select an arbitrary message m and obtain the tags $H_{k_i}(m)$ for $i{=}1,2,\ldots$ until a *collision* is obtained: $H_{k_p}(m){=}H_{k_q}(m)$ where $p < q$. Now corrupt user p and obtain k_p. Under the assumption that $k_p = k_q$, use k_p to forge a message-tag pair with respect to user q. This attack is called *Attack 2.* One can show that if $r \le t$ then the probability that the first collision is a key collision is significant only when the number of MAC queries is at least $2^{r/2}$. Since Attack 1 can succeed with fewer MAC queries, does not require corrupt queries, and is amenable to time-memory trade-offs (cf. Remark 7), it is always superior to Attack 2.

Remark 5. (*symmetric-key encryption*) Attack 1 shows that *existential key recovery* (i.e., finding the secret key of any one of a set of users) is easier than *universal key recovery* (i.e., finding the secret key of a specified user) for (deterministic) MAC schemes; these notions of key recovery were discussed in [49, Section 5] in the context of public-key cryptosystems. Gligor, Parno and Shin (see [70]) proved that existential key recovery is intractable for nonce-based symmetric-key encryption schemes that are indistinguishable against chosen-plaintext attacks; an example of such an encryption scheme is the counter mode of encryption (cf. §5). However, their proof is non-tight, the tightness gap being equal to the number of secret keys in the system. They then showed that this tightness gap allows an existential key recovery attack that is faster than the best attack known for universal key recovery for certain nonce-based encryption schemes including the counter mode of encryption. Attack 1 is analogous to their attack, which in turn was preceded by Biham's key collision attacks [9].

We next argue that Attack 1 is effective on HMAC as standardized in [33, 26] and CMAC as standardized in [28, 72].

HMAC. HMAC [3] is a hash function-based MAC scheme that is extensively standardized and has been widely deployed in practice. The MAC of a message m with secret key k is $\mathrm{HMAC}_k(m) = \mathrm{Trunc}_t\left(H(k \oplus \mathrm{opad}, H(k \oplus \mathrm{ipad}, m))\right)$, where $H : \{0,1\}^* \to \{0,1\}^d$ is an iterated hash function, opad and ipad are fixed strings, and Trunc_t is the truncation function that extracts the t most significant bits of its input. The HMAC parameters are r (the bitlength of the

[3] The case $r \gg t$ can be reduced to the case $t \gg r$ by querying the MAC oracles with s different messages, where s is chosen so that $st \gg r$; see [50, Section 4].

secret key k), t (the bitlength of MAC tags) and d (the output length of H, which is assumed to be an iterated hash function).

IETF RFC 4868 [46] specifies HMAC-SHA-256-128, i.e., HMAC with SHA-256 [32] as the underlying hash function and parameters $r{=}d{=}256$ and $t{=}128$, presumably intended to achieve a 128-bit security level. Since $r \gg t$, Attack 1 is certain to fail when HMAC-SHA-256-128 is used in the multi-user setting.

HMAC is also standardized in FIPS 198-1 [33], with recommendations for parameter sizes given in SP 800-107 [26]. It is stated in [26] that the "security strength" of HMAC is the minimum of r and $2d$, and the only requirement on tag lengths is that $t \geq 8$. Hence, if one were to use HMAC with SHA-1 [32] (which has $d{=}160$) as the underlying hash function and select $r{=}t{=}80$, then the resulting MAC scheme would be compliant with SP 800-107 and be expected to achieve an 80-bit security level. However, this version of HMAC would succumb to Attack 1 in the multi-user setting. Namely, by selecting $n{=}2^{20}$ and $w{=}2^{60}$, after querying 2^{20} users for the MAC of some fixed message m, the adversary would be able to determine the secret key of one of the 2^{20} users after performing about 2^{60} MAC operations. Since the work can be easily and effectively parallelized, the attack should be considered feasible today (cf. Remark 7).

The FIPS 198-1 standard allows 80-bit keys and 160-bit tags, i.e., $r{=}80$ and $t{=}160$. Attack 1 also applies to this choice of parameters. In fact, since $t \gg r$, a collision in the first phase of the attack will most likely be due to a key collision. In general, having tag length to be greater than the key length will not provide any additional resistance to Attack 1.

Remark 6. (*number of users*) The 2^{20} users in the attack described above need not be distinct pairs of entities. What is needed is 2^{20} keys. An entity might be engaged in multiple sessions with other entities, and might even have several active sessions with the same entity. Thus, the attacks could be mounted with far fewer than 2^{20} different entities.

CMAC. CMAC is a block cipher-based MAC scheme that has been standardized in [28] and [72]. Let E denote a block cipher with key length r bits and block length b bits. The r-bit key k is first used to generate two b-bit subkeys, k_1 and k_2. The message m is divided into blocks $m_1, m_2, \ldots, m_h$, where each m_i is b-bits in length with the possible exception of m_h, which might be less than b-bits long. Now, if m_h is b bits in length, then it is updated as follows: $m_h \leftarrow m_h \oplus k_1$. Otherwise, m_h is padded on its right with a single 1 bit followed by 0 bits until the length of the padded m_h is b bits; then m_h is updated as follows: $m_h \leftarrow m_h \oplus k_2$. Finally, one sets $c_0 = 0$ and computes $c_i = E_k(c_{i-1} \oplus m_i)$ for $1 \leq i \leq h$. The tag of m is defined to be $\mathrm{CMAC}_k(m) = \mathrm{Trunc}_t(c_h)$.

The standards [28] and [72] both use the AES block cipher (with $r{=}b{=}128$) and do not mandate truncation, so we can take $t{=}128$. With these parameters, CMAC in the multi-user setting is vulnerable to Attack 1. Indeed, after querying $n{=}2^{32}$ users for the MAC of a fixed message m, the adversary is able to compute the secret key of one of the users after performing about 2^{96} MAC operations. Although this workload is considered infeasible today, the attack does demonstrate that CMAC-AES does not attain the 128-bit security level in the multi-user setting.

Remark 7. (*reducing the on-line running time*) Hellman [39] introduced the idea of time/memory trade-offs (TMTO) to search for a preimage of a target point in the range of a one-way function. The idea is to perform a one-time precomputation and store some of the results, subsequent to which the on-line search phase can be significantly sped up. Biryukov and Shamir [11] later applied TMTO to stream ciphers. They considered the problem of inverting any one out of D possible targets. Let N denote the size of the search space, M the amount of memory required, and T the on-line time, and suppose that $1 \leq D \leq T^2$. Then the Biryukov-Shamir TMTO can be implemented with these parameters provided that they satisfy the so-called multiple-data trade-off curve $TM^2D^2 = N^2$; the precomputation time P is N/D. The multiple-data trade-off curve has natural interpretations in other contexts. Biryukov et al. [10] considered the problem of finding any one of D keys for a block cipher. An extensive analysis of TMTO with multiple data in different cryptographic settings was carried out in [40].

The multiple-data trade-off curve can be applied in the current context to reduce the online search time. For HMAC with $r{=}t{=}80$ as considered above, consider the function $f : k \mapsto \mathrm{HMAC}_k(m)$ where m is a fixed message. Treating f as a one-way function, the adversary's goal is to invert f on any one of the n tag values $f(k_1), \ldots, f(k_n)$. For $n = 2^{20}$, the precomputation time is $P = 2^{60}$ and T and M satisfy $TM^2 = 2^{120}$. Setting $T = M$ (as originally considered by Hellman), we have $T = M = 2^{40}$. Thus, the adversary can find any one of 2^{20} possible HMAC keys with an off-line computation of 2^{60} HMAC invocations, 2^{40} storage units, and an on-line search time of 2^{40}. Using presently available storage and computer technology, this attack should be considered feasible.

For the CMAC example considered above with $r{=}t{=}128$, if the adversary wishes to determine any one of $n = 2^{32}$ possible secret keys, the precomputation time would be $P = 2^{96}$. The parameters T and M are related by $TM^2 = 2^{192}$, so $T = M = 2^{64}$ is one solution. Hence, with 2^{64} storage units, an on-line search time of 2^{64} will find one of 2^{32} keys.

Remark 8. (*two-key and three-key variants of CMAC*) The predecessors of CMAC include a three-key variant called XCBC [12] and a two-key variant called TMAC [52]. Interestingly, these predecessors are not vulnerable to Attack 1 due to the use of multiple keys.

Remark 9. (*comparison with birthday attacks*) HMAC and CMAC are both vulnerable to the following birthday attack in the single-user setting. Suppose that keys and tags are each r bits in length. The adversary collects message-tag pairs (where the messages all have the same length) until two distinct messages m_1 and m_2 are found with the same tag τ. By the birthday paradox, the expected number of pairs needed is approximately $2^{r/2}$. Then, for any string x, (m_1, x) and (m_2, x) have the same tags (with high probability in the case of HMAC, and with certainty in the case of CMAC). The attacker can then request for the tag of (m_1, x), thereby also obtaining the tag of (m_2, x).

Note that Attack 1 can be successful by using signficantly fewer MAC queries, and additionally needs to issue only one MAC query per user. Moreover, the damage caused by Attack 1 is more severe than the birthday attack since the former is a key recovery attack.

2.4 Analysis of Attack 1

The *ideal MAC model* for a MAC scheme $\{H_k : \mathcal{D} \to \{0,1\}^t\}_{k\in\mathcal{K}}$ is the following. Let $\mathcal{F}$ be the set of all functions from $\mathcal{D}$ to $\{0,1\}^t$. The set $\mathcal{F}$ is finite and can be considered as the set of all strings of length $\#\mathcal{D}$ over the alphabet $\{0,1\}^t$. A total of 2^r independent and uniform random choices are made from $\mathcal{F}$, giving a family of 2^r independent random oracles. Each such oracle can be indexed by an r-bit string. The resulting indexed family is the idealized version of a MAC scheme. In what follows, $\{H_k\}_{k\in\mathcal{K}}$ will denote an idealized MAC family. In particular, the H_k's will be considered to be independent uniform random oracles.

Consider the following procedure. Suppose $k_1, \dots, k_n$ are chosen independently and uniformly at random from $\mathcal{K} = \{0,1\}^r$. Let m (a message) be an arbitrary element of $\mathcal{D}$. Then, for $i \neq j$, we will need to consider the event that $H_{k_i}(m) = H_{k_j}(m)$. For the probability analysis, it will be useful to analyze this event in terms of the following three events, the last two of which are conditional events: (i) $k_i = k_j$; (ii) $H_{k_i} = H_{k_j}$ given that $k_i \neq k_j$; and (iii) $H_{k_i}(m) = H_{k_j}(m)$ given that $k_i \neq k_j$ and $H_{k_i} \neq H_{k_j}$. Clearly $\Pr[k_i = k_j] = 2^{-r}$ and $\Pr[H_{k_i} = H_{k_j} | k_i \neq k_j] = 1/\#\mathcal{F} = (2^{-t})^{\#\mathcal{D}}$. In practical applications, the maximum length L of messages can be expected to be at least around 2^{20} and so the probability that $H_{k_i} = H_{k_j}$ given $k_i \neq k_j$ is negligible. Furthermore, for $1 \leq s \leq 2^t$, the quantity $s/\#\mathcal{F}$ is also negligible. We will use these approximations in the remainder of the analysis.

The analysis of Attack 1 is done in two stages. In the first stage, we determine values for n and w for which there is a significant probability of detecting a collision. The second stage of the analysis considers the probability of the keys ℓ and k_i being equal once a collision $H_\ell(m) = H_{k_i}(m)$ is detected.

Let $\mathcal{W} = \{\ell_1, \dots, \ell_w\}$ and consider the functions $H_{\ell_1}, \dots, H_{\ell_w}$. Let A be the event that these functions are pairwise distinct. Then

$$\Pr[A] \;=\; \left(1 - \frac{1}{\#\mathcal{F}}\right)\left(1 - \frac{2}{\#\mathcal{F}}\right)\cdots\left(1 - \frac{w-1}{\#\mathcal{F}}\right) \;\approx\; 1.$$

The approximation is based on the fact that w^2 is negligible in comparison to $\#\mathcal{F} = (2^t)^{\#\mathcal{D}}$. Let C be the event that a collision occurs. Let $\mathsf{Lst}_1 = \{H_{k_1}(m), \dots, H_{k_n}(m)\}$ and $\mathsf{Lst}_2 = \{H_{\ell_1}(m), \dots, H_{\ell_w}(m)\}$. The event C is the event $\mathsf{Lst}_1 \cap \mathsf{Lst}_2 \neq \emptyset$. Now,

$$\Pr[C] \;=\; \Pr[C|A] \cdot \Pr[A] + \Pr[C|\overline{A}] \cdot \Pr[\overline{A}] \;\approx\; \Pr[C|A].$$

Let B_1 be the event that the keys $k_1, \ldots, k_n$ are pairwise distinct. Then

$$\begin{aligned}\Pr[B_1] = \left(1 - \frac{1}{2^r}\right) \cdots \left(1 - \frac{n-1}{2^r}\right) \;\approx\; \exp\left(-\frac{1}{2^r}(1 + 2 + \cdots + n - 1)\right) \\ \approx\; \exp\left(-\frac{n^2}{2^{r+1}}\right) \;\approx\; 1 - \frac{n^2}{2^{r+1}}.\end{aligned}$$

As long as $n^2 \ll 2^{r+1}$, the probability of event B_1 occurring will be almost equal to 1. For the remainder of the analysis, we will assume that this condition holds.

Let B_2 be the event that the functions $H_{k_1}, \ldots, H_{k_n}$ are pairwise distinct. Conditioned on the event B_1, the probability of B_2 occurring is almost equal 1. This follows from an argument similar to the one which shows that $\Pr[A] \approx 1$. We introduce three more approximations:

$$\begin{aligned}\Pr[C] \;\approx\; \Pr[C|A] &= \Pr[C|A, B_1] \cdot \Pr[B_1] + \Pr[C|A, \overline{B_1}] \cdot \Pr[\overline{B_1}] \\ &\approx \Pr[C|A, B_1] \qquad (\text{using } \Pr[B_1] \approx 1) \\ &= \Pr[C|A, B_1, B_2] \cdot \Pr[B_2] + \Pr[C|A, B_1, \overline{B_2}] \cdot \Pr[\overline{B_2}] \\ &\approx \Pr[C|A, B_1, B_2] \qquad (\text{using } \Pr[B_2] \approx 1).\end{aligned}$$

Let $x_i = H_{k_i}(m)$ for $1 \le i \le n$ and $y_j = H_{\ell_j}(m)$ for $1 \le j \le w$. Conditioned on the conjunction of B_1 and B_2, the values $x_1, \ldots, x_n$ are independent and uniformly distributed. Conditioned on event A, the values $y_1, \ldots, y_w$ are independent and uniformly distributed. Hence, conditioned on the conjunction of A, B_1 and B_2, and assuming that the probability of H_{k_i} being equal to H_{ℓ_j} for any i, j is negligible, the event C is the event that a list of n independent and uniform values from $\{0, 1\}^t$ has a non-empty intersection with another list of w independent and uniform values from $\{0, 1\}^t$. By the birthday bound, this probability becomes significant when the product $n \cdot w$ is some constant times 2^t. As an example, one may choose $n = 2^{t/4}$ and w to be a constant times $2^{3t/4}$. Similarly, conditioned on the conjunction of A, B_1 and B_2, and assuming that the probability of a tag collision occurring is negligible, the probability of event C becomes significant when $n \cdot w$ is some constant times 2^r.

Suppose now that a collision is detected. The probability that the collision is due to a repetition of the keys can be estimated as follows. We have

$$\begin{aligned}\Pr[H_{k_i}(m) = H_{\ell_j}(m)] &= \Pr[k_i = \ell_j] + \Pr[H_{k_i}(m) = H_{\ell_j}(m) | k_i \neq \ell_j] \cdot \Pr[k_i \neq \ell_j] \\ &= \frac{1}{2^r} + \left(1 - \frac{1}{2^r}\right) \cdot \Bigg(\Pr[H_{k_i} = H_{\ell_j} | k_i \neq \ell_j] \\ &+ \Pr[H_{k_i}(m) = H_{\ell_j}(m) | k_i \neq \ell_j, H_{k_i} \neq H_{\ell_j}] \cdot \Pr[H_{k_i} \neq H_{\ell_j} | k_i \neq \ell_j] \Bigg) \\ &= \frac{1}{2^r} + \left(1 - \frac{1}{2^r}\right) \left(\frac{1}{\#\mathcal{F}} + \frac{1}{2^t} \left(1 - \frac{1}{\#\mathcal{F}}\right)\right) \approx \frac{1}{2^r} + \frac{1}{2^t} - \frac{1}{2^{t+r}} = \delta,\end{aligned}$$

and hence

$$\Pr[k_i = \ell_j | H_{k_i}(m) = H_{\ell_j}(m)] = \frac{\Pr[k_i = \ell_j, H_{k_i}(m) = H_{\ell_j}(m)]}{\Pr[H_{k_i}(m) = H_{\ell_j}(m)]}$$
$$= \frac{\Pr[k_i = \ell_j]}{\Pr[H_{k_i}(m) = H_{\ell_j}(m)]} \approx \frac{1}{2^r\delta} = \frac{2^{t+r}}{2^r(2^t + 2^r - 1)} = \frac{2^t}{2^t + 2^r - 1}.$$

If $r = t$, then the last value is approximately $\frac{1}{2}$. However, if $r \gg t$, then the probability is essentially 0. If $r \ll t$, the probability is essentially 1.

2.5 Fixes

We propose two generic countermeasures to Attack 1 on MAC schemes in the multi-user setting.

Remark 10. (*preventing replay attacks*) Some MAC standards make provisions for protecting against the replay of message-tag pairs. For example, NIST's SP 800-38B [28] suggests that replay can be prevented by "incorporating certain identifying information bits into the initial bits of every message. Examples of such information include a sequential message number, a timestamp, or a nonce." We note that sequential message numbers and timestamps do not necessarily circumvent Attack 1 because it is possible that each user selects the same sequential message number or timestamp when authenticating the chosen message m. Nonces can be an effective countermeasure provided that there is sufficient uncertainty in their selection.

rMAC. One countermeasure is to randomize the conventional MAC scheme $\{H_k\}_{k\in\mathcal{K}}$. That is, a user with secret key k now authenticates a message m by computing $\tau = H_k(s, m)$ where $s \in_R \{0,1\}^r$; the resulting tag is (s, τ). The verifier confirms that $\tau = H_k(s, m)$. This modified MAC scheme is called *rMAC* (randomized MAC).

Security of rMAC in the multi-user setting is defined analogously to security of MAC*: The adversary $\mathcal{A}$ is given access to n rMAC oracles with secret keys $k_1, k_2, \ldots, k_n \in_R \mathcal{K}$ and can corrupt any oracle (i.e., obtain its secret key). Its goal is to produce a triple $(i, m, (s, \tau))$ such that the ith oracle was not corrupted, $(m, (s, \tau))$ is a valid message-tag pair with respect to the ith oracle (i.e, $H_{k_i}(s, m) = \tau$), and m was not queried to the ith oracle. We denote $\mathcal{A}$'s task by *rMAC**. When $n = 1$, then rMAC* is called *rMAC1* (security of rMAC in the single-user setting).

It is easy to verify that rMAC* resists Attack 1. Let us denote by $\mathcal{P}_1 \leq_b \mathcal{P}_2$ a reduction from problem $\mathcal{P}_1$ to problem $\mathcal{P}_2$ that has a tightness gap of b; if $b = 1$ then the reduction is tight. In §2.2 we showed that MAC1 $\leq_n$ MAC*, i.e., the problem of breaking a MAC scheme in the single-user setting can be reduced to breaking the same MAC scheme in the multi-user setting, but the reduction has a tightness gap of n. Trivially, we have MAC* $\leq_1$ MAC1. The reductionist security proof in §2.2 can be adapted to show that rMAC1 $\leq_n$ rMAC*, and we trivially have rMAC* $\leq_1$ rMAC1. Moreover, it is easy to see that MAC1 $\leq_1$ rMAC1 and hence MAC1 $\leq_n$ rMAC*, and that MAC* $\leq_1$ rMAC*. However, it is unlikely that a generic reduction of rMAC1 to MAC1 exists because a MAC scheme $\{H_k\}_{k\in\mathcal{K}}$ having the property that there exists a (known) pair (σ,τ) with $\sigma \in \{0,1\}^r$, $\tau \in \{0,1\}^t$ and $H_k(\sigma) = \tau$ for all $k \in \mathcal{K}$ would be considered insecure whereas the corresponding rMAC scheme could well be secure.

We do not know a tighter security reduction from MAC1 to rMAC*, nor do we know whether a tighter reduction is even possible (in general). However, we would expect that rMAC* and MAC1 are tightly related in practice. One approach to increasing confidence in rMAC* would be to derive tight lower bounds for MAC1 and rMAC* in the ideal MAC model, and hope that these lower bounds coincide.

fMAC. One drawback of rMAC is that tags are longer than before. An alternative countermeasure is to prepend all messages with a string that is fixed and unique to every pair of users (and every session between them). That is, a user with secret key k would authenticate a message m by computing the tag $\tau = H_k(f,m)$, where f is the fixed and unique string that the user shares with the intended recipient (for that session). All such strings are assumed to have the same length, and this length is at least r. The strings are assumed to be understood from context, so do not need to be transmitted. (For an example of such strings, see §3.3.) The verifier confirms that $\tau = H_k(f,m)$. This modified MAC scheme is called *fMAC* (fixed-string MAC).

Security of fMAC in the multi-user setting is defined analogously to security of MAC*: The adversary $\mathcal{A}$ is given access to n fMAC oracles with secret keys $k_1,\ldots,k_n \in_R \mathcal{K}$ and fixed strings $f_1,\ldots,f_n$, and can corrupt any oracle. Its goal is to produce a triple (i,m,τ) such that the ith oracle was not corrupted, (m,τ) is a valid message-tag pair with respect to the ith oracle (i.e, $H_{k_i}(f_i,m) = \tau$), and m was not queried to the ith oracle. We denote $\mathcal{A}$'s task by *fMAC**. When $n = 1$, then fMAC* is called *fMAC1* (security of fMAC in the single-user setting).

As was the case with rMAC*, it is easy to verify that fMAC* resists Attack 1. Furthermore, one can show that fMAC* $\leq_1$ fMAC1, MAC1 $\leq_1$ fMAC1 $\leq_n$ fMAC*, and MAC* $\leq_1$ fMAC*, while we do not expect there to be a generic reduction from fMAC1 to MAC1. We do not know a tighter security reduction from MAC1 to fMAC*, nor do we know whether a tighter reduction is even possible (in general). However, we would expect that fMAC* and MAC1 are tightly related in practice. An intuitive reason for why fMAC* can be expected to be more secure than MAC* is that for fMAC* each of the n oracles available to the adversary can be viewed as having been chosen from an *independent* family of MAC functions, whereas in MAC* each of the n oracles available to the adversary is chosen from a *single* family of MAC functions.

Remark 11. (*use of MAC schemes*) Higher-level protocols that use MAC schemes for authentication generally include various data fields with the messages being MAC'ed, thus providing adequate defenses against Attack 1. For example, IPsec has an authentication-only mode [47] where a MAC scheme is used to authenticate the data in an IP packet. Among these data fields are the source and destination IP addresses, and a 32-bit "Security Parameter Index" (SPI) which identifies the "Security Association" (SA) of the sending party.

3 NetAut

NetAut is a network authentication protocol proposed by Canetti and Krawczyk [20] which combines a key establishment scheme with a conventional MAC scheme in a natural way. In [20], a security model and definition for key establishment are proposed. Then, NetAut is proved to be a secure network authentication protocol under the assumption that the underlying key establishment and MAC schemes are secure. We describe several shortcomings in the analysis of NetAut. The most serious of these shortcomings is the tightness gap in the security proof, which we exploit to formulate concrete attacks on plausible instantiations of NetAut.

3.1 Network authentication

The NetAut protocol presented in [20] has two ingredients: a key establishment protocol π and a MAC scheme. NetAut utilizes a *session identifier* s, which is a string agreed upon by the parties before execution of the protocol commences. It is assumed that no two NetAut sessions in which a party $\hat{A}$ participates with another party $\hat{B}$ have the same session identifier.

In the initial stage of the NetAut protocol[4], a party $\hat{A}$ participates in a key establishment session with another party $\hat{B}$. Upon successful completion of the session, $\hat{A}$ accepts a session key κ associated with the session identified by s and (presumably) shared with $\hat{B}$. Now, to send $\hat{B}$ an authenticated message m within session s, $\hat{A}$ computes $\tau = \mathrm{MAC}_\kappa(m)$ and sends $(\hat{A}, s, m, \tau)$ to $\hat{B}$. Similarly, upon receipt of a message $(\hat{B}, s, m, \tau)$, $\hat{A}$ computes $\tau' = \mathrm{MAC}_\kappa(m)$ and accepts if $\tau' = \tau$. At any point in time, $\hat{A}$ can have multiple active sessions, and can even have multiple active sessions with $\hat{B}$.

The security model for NetAut is developed in two stages. The first stage defines what it means for a key establishment protocol to be secure. The security model, which has come to be known as the 'CK model', allows for multiple parties and multiple sessions, and gives the adversary substantial powers including the ability to learn some session keys, corrupt parties (learn all their secret information), and learn some secret information that is specific to a particular session. Informally speaking, a key establishment protocol is said to be secure if no such adversary can distinguish the session key held by a fresh session from a randomly-generated session key, where a 'fresh' session is one for which the adversary cannot learn the corresponding session key through trivial means (such as corrupting the party that participates in that session or simply asking for the session key). A crucial feature of the definition is that any key establishment

[4] Our description of NetAut is informal and omits a lot of details; the reader can refer to [20] for a complete description.

protocol that satisfies the security definition can be appropriately combined with secure MAC and symmetric-key encryption schemes to realize a 'secure channel'. In this paper, we will only consider NetAut — the combination of a key establishment protocol with a MAC scheme.

The second stage of the security model for NetAut starts with an idealized notion called a session-based message transmission (SMT) protocol in the authenticated links model. In the authenticated links model, the communications links between any two parties is perfectly authenticated — the SMT protocol is secure in this model by its very definition. A secure network authentication protocol is then defined as one that 'emulates' SMT in the unauthenticated links model in the sense that whatever an adversary can achieve against the protocol can also be accomplished by an adversary against SMT in the authenticated links model.

Canetti and Krawczyk prove that if π is a secure key establishment protocol and the MAC scheme is secure (in the single-user setting), then NetAut is a secure network authentication protocol. The proof and associated definitions are long and intricate. In §3.2 we describe some pitfalls that arise in interpreting the proof when NetAut is instantiated with the SIG-DH key establishment protocol.

3.2 A concrete analysis

For concreteness, we will consider the 80-bit security level. Let E be an elliptic curve defined over $\mathbb{F}_p$ where p is a 160-bit prime. Suppose that $N = \#E(\mathbb{F}_p)$ is prime, so that the group $E(\mathbb{F}_p)$ of $\mathbb{F}_p$-rational points offers an 80-bit security level against attacks on the discrete logarithm problem. Let G be a fixed generator of $E(\mathbb{F}_p)$. We consider CMAC at the 80-bit security level, i.e., with 80-bit keys and 80-bit tags; the block cipher SKIPJACK [59], which has 80-bit keys and 80-bit blocks, is a suitable ingredient.

In the SIG-DH key agreement scheme, $\mathrm{sig}_{\hat{A}}$ and $\mathrm{sig}_{\hat{B}}$ denote the signing algorithms of parties $\hat{A}$ and $\hat{B}$, respectively. It is assumed that each party has an authenticated copy of the other party's public verification key. The SIG-DH scheme proceeds as follows. The initiator $\hat{A}$ selects $x \in_R [0, N-1]$ and sends $(\hat{A}, s, X{=}xG)$ to party $\hat{B}$. In response, $\hat{B}$ selects $y \in_R [0, N-1]$ and sends $(\hat{B}, s, Y{=}yG, \mathrm{sig}_{\hat{B}}(\hat{B}, s, Y, X, \hat{A}))$ to $\hat{A}$ and computes $\kappa = yX$. Upon receipt of $\hat{B}$'s message, $\hat{A}$ verifies the signature, sends the message $(\hat{A}, s, \mathrm{sig}_{\hat{A}}(\hat{A}, s, X, Y, \hat{B}))$ to $\hat{B}$, and computes the session key $\kappa = xY$ associated with session s. Finally, upon receipt of $\hat{A}$'s message, $\hat{B}$ verifies the signature and accepts κ as the session key associated with session s.

Canetti and Krawzcyk proved that SIG-DH is secure in the CK model under the assumption that the decisional Diffie-Hellman problem[5] in $E(\mathbb{F}_p)$ is intractable (and the signature scheme is secure). The proof proceeds in two stages. In the first stage, the basic Diffie-Hellman protocol is proven secure in the authenticated links model under the assumption that DDH is intractable; this proof

[5] The decisional Diffie-Hellman (DDH) problem in $E(\mathbb{F}_p)$ is the problem of determining whether $Z{=}xyG$ given G, $X{=}xG$, $Y{=}yG$ and $Z \in E(\mathbb{F}_p)$.

has a tightness gap of n, the total number of sessions. In the second stage, SIG-DH is proven secure (in the unauthenticated links model) under the assumption that the basic Diffie-Hellman protocol is secure in the authenticated links model; this proof has a tightness gap of $2n$. However, these tightness gaps do not seem to have any negative security consequences for SIG-DH.

Key type mismatch. The first problem encountered when using SIG-DH and CMAC as the ingredients of NetAut is that the SIG-DH session keys are points in $E(\mathbb{F}_p)$ whereas the CMAC secret keys are bit strings. This *key type mismatch* can be rectified by the commonly-used method of using a key derivation function KDF to derive a bit-string session key from the SIG-DH session key, i.e., the session key is now KDF(xyG). We refer to the modified key agreement scheme as *hashed SIG-DH* (HSIG-DH).

The KDF is generally modeled as a random oracle in security proofs. HSIG-DH can then be proven secure under the assumption that the gap Diffie-Hellman (GDH) problem[6] is hard using standard techniques.

Keysize mismatch. Security proofs for Diffie-Hellman key agreement protocols in the random oracle model sometimes make the assumption that the probability of a KDF collision during the adversary's operation is negligible (e.g., see [51, 53, 58]). If this probability were not negligible, then the adversary could conceivably force two non-related sessions (called 'non-matching' sessions in the literature) to compute the same session key — in that event, the adversary could learn the session key from one session by asking for it and thereby obtain the session key for the other session. Thus, because of the birthday paradox, at the 80-bit security level the assumption that the adversary has negligible probability of obtaining a KDF collision requires that the KDF for HSIG-DH with our choice of elliptic curve parameters should have 160-bit outputs. However we then have a *keysize mismatch* since CMAC uses 80-bit keys. If the KDF is restricted to 80-bit outputs, then the aforementioned proofs have a logical gap since the probability of a KDF collision now becomes non-negligible.

One simple way to remove this gap is to include the identities of the communicating parties and the session identifier as input to the key derivation function (as is done in [73], for example), i.e., the HSIG-DH session key is now KDF($\hat{A}, \hat{B}, s, xyG$). One can then argue that since the KDF is modelled as a random oracle, the adversary *must* know the inputs to the KDF for the two non-matching sessions (since the triples $(\hat{A}, \hat{B}, s)$ for the non-matching sessions must be distinct) in order to detect the collision. In particular, the adversary must know xyG — and such an adversary can be used to solve a CDH instance.

The insecurity of NetAut. Attack 1 is applicable to our instantiation of NetAut with HSIG-DH (with 80-bit session keys) and CMAC at the 80-bit security level. Namely, the adversary monitors $n = 2^{20}$ NetAut sessions, each of which is induced to transmit some fixed message m. Then, as explained in §2.3, the

[6] The gap Diffie-Hellman (GDH) problem in $E(\mathbb{F}_p)$ is the problem of solving the computational Diffie-Hellman (CDH) problem in $E(\mathbb{F}_p)$ given an oracle for the DDH problem in $E(\mathbb{F}_p)$.

adversary is able to deduce one of the 2^{20} session keys and thereafter use it to forge message-MAC pairs for that session.

We emphasize that the mechanisms of the attack are within the scope of the security model for NetAut considered in [20]. However, the attack does not contradict the security proof for NetAut given in [20, Theorem 12] for the following reason. At one point in the proof it is shown that the probability that an adversary succeeds in convincing a party $\hat{A}$ that a message m was sent by party $\hat{B}$ in a particular session s even though $\hat{B}$ did not send that message in that session is negligible provided that the underlying MAC scheme is secure. The reductionist proof for this claim (Lemma 13 of [20]) is analogous to the security proof for MAC* given in §2.2, and hence has a tightness gap equal to the total number n of sessions — this tightness gap is precisely what the attack exploits.

3.3 A fix

One method for preventing the attack on NetAut described above is to use the fMAC variant of the MAC scheme. Here, a natural candidate for the unique fixed string f is the session identifier s and the identifiers of the communicating parties, i.e., after parties $\hat{A}$ and $\hat{B}$ complete session s of HSIG-DH and establish a session key κ, the authentication tag for a message m is computed as $\tau = \text{MAC}_\kappa(s, \hat{A}, \hat{B}, m)$. This modification of NetAut resists Attack 1. However, even with this modification we do not know a tight security reduction, so the possibility of another attack that exploits the tightness gap cannot be ruled out.

4 Aggregate MAC schemes

In the section, we show that some aggregate MAC schemes with non-tight security proofs and an aggregate designated verifier signature are vulnerable to Attack 1 for certain choices of the underlying MAC scheme, e.g., CMAC with 80-bit keys and 80-bit tags.

4.1 Aggregate MAC schemes

Katz and Lindell [44] provided a formal security definition for the task of aggregating MACs, proposed an aggregate MAC scheme, and gave a security proof for their construction.

In the Katz-Lindell scheme, there are z parties, each of which randomly selects an r-bit key k_i for a deterministic MAC scheme; these keys are shared with a central authority. When parties[7] $i_1, i_2, \ldots, i_n$ wish to authenticate messages $m_1, m_2, \ldots, m_n$, respectively, for the authority, they each compute $\tau_i = \text{MAC}_{k_i}(m_i)$. The *aggregate tag* is $\tau = \tau_1 \oplus \tau_2 \oplus \cdots \oplus \tau_n$. The authority verifies the aggregate tag by computing the individual tags and checking that their xor is equal to τ.

[7] For simplicity, we assume the parties are distinct and hence $n \leq z$.

In the security model of [44], the adversary can corrupt any party, and in addition can obtain the tag of any message from any party. The adversary's goal is to produce a set of party-message pairs $(i_1, m_1), (i_2, m_2), \ldots, (i_n, m_n)$ (for any $n \leq z$) and an aggregate tag τ such that the tag passes the verification check and there is at least one party-message pair (i_j, m_j) for which party i_j has not been corrupted and was never queried for the MAC of m_j.

Katz and Lindell prove that their aggregate MAC scheme is secure provided that the underlying MAC scheme is secure in the single-user setting. Their proof is very similar to the one given for MAC* in §2.2, but is described asymptotically. The total number of parties is $z = p(r)$ for some unspecified polynomial p, and the adversary $\mathcal{A}$ of the aggregate MAC scheme is assumed to be polynomially bounded. The simulator $\mathcal{B}$ of $\mathcal{A}$'s environment makes a guess for the index j, and is successful in producing a forgery for the underlying MAC scheme provided that its guess is correct. Since $n \leq z$, the proof has a tightness gap of $p(r)$.

It is easy to see that the Katz-Lindell aggregate MAC scheme succumbs to Attack 1. This security flaw in their scheme is a direct consequence of the tightness gap in their proof.

As with rMAC, randomizing the MACs will prevent the attack. However, since the randomizers would also have to be sent, this countermeasure defeats the primary objective of the aggregate MAC scheme — a small aggregate tag. A better solution would be to deploy fMAC as the underlying MAC scheme.

Hierarchical in-network data aggregation. Chan, Perrig and Song [21] presented the first provably secure hierarchical in-network data aggregation algorithm. Such an algorithm can be used to securely perform queries on sensor network data. A crucial component of the algorithm is the (independently discovered) Katz-Lindell aggregate MAC scheme. In the data aggregation application, each sensor node shares a secret key k_i with the querier. At one stage of the application, each node computes the tag $\tau_i = \mathrm{MAC}_{k_i}(N, OK)$, where MAC is a conventional MAC scheme, N is a nonce sent by the querier, and OK is a unique message identifier. The aggregate tag is $\tau = \tau_1 \oplus \tau_2 \oplus \cdots \oplus \tau_n$. We emphasize that the *same* nonce N and message identifier OK are used by each node. It follows that the MAC scheme is vulnerable to Attack 1. In fact, the attack is easier to mount in this setting because the application itself requires each node to compute its tag on a fixed message. The security proof for the aggregate MAC scheme given in [21, Lemma 11] is very informal and assumes "that each of the distinct MACs are unforgeable (and not correlated with each other)", and then concludes that "the adversary has no information about this [aggregate tag]."

History-free aggregate MACs. Eikemeier et al. [31] presented and analyzed a MAC aggregation algorithm where the aggregation of individual tags must be carried out in a sequential manner, and where the aggregation algorithm depends only on the current message being MAC'ed and on the previous aggregate tag. They provided an elaborate security definition and a security proof for their scheme. We note that their security model allows the adversary to query individual parties for tags of messages of the adversary's choosing. Consequently, their history-free aggregate MAC scheme succumbs to Attack 1. Not surprisingly, the security reduction in [31] is non-tight, with a tightness gap of at least z (the total number of parties).

4.2 Aggregate designated verifier signatures

An aggregate designated verifier signature (ADVS) scheme combines the ideas of aggregate signatures [17] and designated verifier signatures [41]. Bhaskar, Herranz and Laguillaumie [8] introduced the notion of ADVS and proposed two constructions in the public-key and identity-based settings. The constructions at their core use a MAC scheme and the identical idea of MAC aggregation as in Katz-Lindell (§4.1). The essential difference is that the common MAC key of a sender and the designated verifier is derived from the discrete-log static keys of the two parties through hashing.

It is easy to see that the Bhaskar et al. scheme is vulnerable to Attack 1. Such an attack, though realistic, is not captured in the security model of [8] which is essentially an adaptation of the aggregate signature security model of Boneh et al. [17]. In particular, both models fail to capture the scenario where multiple honest signers send individual as well as aggregated authenticated messages to a designated verifier, and an adversary is trying to forge a non-trivial (aggregate) signature involving at least one honest signer.

5 Symmetric-key encryption in the multi-user setting

Bellare, Boldyreva and Micali [2] proved that if a public-key encryption scheme is secure in the single-user setting, then it is also secure in the multi-user setting. Their security proof has a tightness gap equal to nq_e, where n is the number of users and q_e is the number of encryptions performed by each user. They mention that analogous results for symmetric-key encryption schemes can be easily proven. In this section, we examine the security of authenticated encryption (AE) schemes and stream ciphers in the multi-user setting.

5.1 Deterministic authenticated encryption

Rogaway and Shrimpton [65] proposed the notion of ‘deterministic authenticated encryption’ (DAE), presented a DAE scheme called Synthetic Initialization Vector (SIV), and proved the scheme secure. A primary motivation for their work was that prior protocols for the ‘key-wrap problem’ had “never received a provable-security treatment”.

Let E be a block cipher with r-bit keys and r-bit blocks. The SIV mode of operation described in [65] uses CMAC and the counter (CTR) mode of operation for E [27]; recall that in CTR mode encryption, a one-time pad is generated by selecting a random IV which is repeatedly incremented and encrypted; the one-time pad is then xored with the blocks of the plaintext to obtain the ciphertext[8]. A plaintext message m is processed by first computing $IV = \mathrm{CMAC}_{k'}(m)$ and then $c = \mathrm{CTR}_{k''}(IV, m)$. Here, the secret key is $k = (k', k'')$, where k' is a key for CMAC and k'' is a key for the block cipher E. The ciphertext is (IV, c). To decrypt and verify, one computes $m = \mathrm{CTR}_{k''}(IV, c)$ and verifies that $IV = \mathrm{CMAC}_{k'}(m)$.

An attack. For concreteness, suppose that SIV uses an 80-bit block cipher (such as SKIPJACK) as the underlying block cipher for CTR mode encryption as well as for CMAC. Our attack on SIV is a chosen-plaintext attack in the multi-user setting. The adversary selects an arbitrary message m and obtains the ciphertext (IV_i, c_i) from 2^{20} parties i with secret key pairs $k_i = (k'_i, k''_i)$. As in Attack 1, the adversary then finds k'_j for some user j in about 2^{60} steps. Next, the adversary finds two equal-length messages m_1 and m_2 with $m_1 \neq m_2$ and $\mathrm{CMAC}_{k'_j}(m_1) = \mathrm{CMAC}_{k'_j}(m_2)$; this can be accomplished in about 2^{40} steps using the van Oorschot-Wiener collision finding algorithm [74]. Finally, the adversary requests the encryption of m_1 from party j, receiving the ciphertext (IV_1, c_1). The adversary then computes the encryption of m_2 as $(IV_1, c_1 \oplus m_1 \oplus m_2)$ as its forgery. It can easily be checked that this ciphertext will decrypt to m_2 and pass the verification check.

Our attack shows that, despite the provable security guarantees of SIV in [65], this particular implementation of SIV does not achieve the desired 80-bit security level in the multi-user setting. Note, however, that the attack may not be relevant in the context of the key-wrap problem. Since "the plaintext carries a key", it will not be possible for the adversary to obtain 2^{20} (IV_i, c_i) pairs on the *same* message m.

A fix. A possible countermeasure to the attack would be to encrypt IV under k'', i.e., the encryption of m would be $(E_{k''}(IV), \mathrm{CTR}_{k''}(IV, m))$ where $IV = \mathrm{CMAC}_{k'}(m)$.

5.2 Authenticated encryption

In many AE schemes, including OCB [64, 62], GCM [56] and PAE [66], the encryption function uses a secret key k for a block cipher to map a nonce-message pair (IV, m) to a ciphertext of the form (c, τ). For these AE schemes, the only requirement on the IV is that it not be repeated with the same key. We consider the scenario where keys, tags and blocks all have the same length.

An attack. Fix a nonce-message pair (IV, m) and consider the function $f : k \mapsto \tau$, where τ is the tag of the AE encryption of (IV, m) under key k. Attack 1 can then be mounted (cf. Remark 7). The attack requires many users to perform authenticated encryption of m with the fixed IV, but since the AE schemes only

[8] In the interest of simplicity, our description of SIV omits some details from [65]. In particular, we omit the header which in any case "may be absent", and use CMAC instead of CMAC*. These omissions do not have any bearing on our attack.

mandate that the IV not be repeated with the same key, the attack is legitimate in the multi-user setting.

Rogaway [63], on his web page that promotes OCB, states

> In the past, one had to wait years before using a new cryptographic scheme; one needed to give cryptanalysts a fair chance to attack the thing. Assurance in a scheme's correctness sprang from the *absence* of damaging attacks by smart people, so you needed to wait long enough that at least a few smart people would try, and fail, to find a damaging attack. But this entire approach has become largely outmoded, for schemes that are not true primitives, by the advent of provable security. With a provably-secure scheme assurance does not stem from a failure to find attacks; it comes from proofs, with their associated bounds and definitions.

In particular, he states that for OCB "the underlying definition is simple and rock solid". It is understandable that practitioners would be glad to hear the recommendation that they can have confidence in a newly proposed protocol solely based on the security proof, and need not wait for it to stand the test of time. However, our attack on OCB, which is a practical one under certain plausible assumptions, shows that it would be more prudent not to put all one's trust in a reductionist security proof and its associated definition, especially if the proof has a large tightness gap or the definition does not allow for the multi-user setting.

5.3 Disk encryption

A disk encryption scheme is a special case of a tweakable enciphering scheme (TES) [37] where the message length is fixed. More concretely, a message is a disk sector and there is a 'tweak' which is the sector address. The tweak is not a nonce in the sense that it can be reused for encryptions with the same key. Formally, the encryption algorithm uses a secret key k to transform a tweak-message pair (IV, m) to a ciphertext c, where c and m have the same length.

For disk encryption schemes such as EME [38], k is a key of a block cipher. By treating c as a tag, one can apply Attack 1 to recover k. Note that since c will typically be much longer than k, a collision encountered during the attack will most likely be due to a key collision. In the context of disk encryption, there is no notion of session keys — the different keys would correspond to different users. The encryption of a fixed tweak-message pair can be obtained by inducing the users to encrypt the chosen message for the chosen disk sector.

Fixes for AE and disk encryption schemes. In the multi-user setting, one way to ensure that an r-bit security level is achieved against our attacks (without changing the underlying block cipher) is to use multiple keys that together are longer than r bits. Examples of such schemes are Poly1305-AES [6] and the disk encryption schemes in [67]. The use of multiple keys, however, does not immediately guarantee resistance to Attack 1 — as we have seen, SIV is vulnerable to the attack since the first ciphertext component depends only on the first SIV key – and hence the modification of a mode of operation to resist Attack 1 should be done with care.

5.4 Stream ciphers

A stream cipher with IV takes as input an r-bit key k and a v-bit IV and produces a keystream which is then XORed with the message to obtain the ciphertext. The usual requirement on the IV is that it should not be repeated for the same key.

Fix a value IV_0 for the IV and define a map f that takes k to the first r bits of the keystream produced using k and IV_0. In the multi-user setting, Attack 1 can be mounted by inducing different users to encrypt known messages using IV_0 and their respective keys. Inverting f on any of the resulting keystreams yields one of the secret keys. For concreteness, consider 80-bit keys and suppose that the attacker is able to collect 2^{20} targets. A TMTO attack using a precomputation of 2^{60} and memory and on-line time of 2^{40} will (with high probability) find one of the 2^{20} keys. The attack parameters are feasible, thus bringing into question the adequacy of 80-bit keys for stream ciphers with IV. The importance of this issue can be seen in the context of the eSTREAM project [30] which recommends 80-bit stream ciphers such as Trivium.

Requiring that IVs be randomly generated does not circumvent the attack but instead makes it somewhat easier to mount. This is because random IVs must be communicated in the clear to a receiver. The attacker could then target the receiver and obtain the first 80 bits of the keystream produced by the receiver. Since a receiver expects IVs along with the ciphertext, an active attacker can legitimately use the same IV_0 for all the 2^{20} receivers. In contrast, if the IV is merely a nonce (such as a counter), then it may be more difficult to induce all senders to use IV_0. Note that the use of an authenticated encryption scheme together with random IVs foils the attack. The attack can also be foiled by using the technique employed in fMAC — prepending the IV with a string that is fixed and unique among all sessions.

6 Concluding remarks

We showed that ignoring the tightness gaps in reductionist security proofs can have damaging consequences in practice. Our examples involve MAC schemes in the multi-user setting. In particular, the tightness gap in the natural reduction from MAC1 to MAC* indicates a real security weakness, whereas the tightness gap in the natural reductions from MAC1 to rMAC* and fMAC* do not seem to matter in practice. Our examples illustrate the difficulty of interpreting a non-tight security proof in practice. Although our examples all involve the multi-user setting, we feel that they call into question the practical value of *all* non-tight security proofs. We also demonstrated potential security weaknesses of provably-secure authenticated encryption schemes in the multi-user setting.

Practitioners who use security proofs as a tool to assess the security of a cryptographic system, but rely more heavily on extensive cryptanalysis and sound engineering principles, should not be alarmed by our observations. On the other hand, theoreticians who believe that a security proof is the essential, and perhaps the *only*, way to gain confidence in the security of a protocol should be much more skeptical of non-tight proofs (unless, of course, the proof is accompanied by a clearly-stated requirement that security parameters be increased to accommodate the tightness gap) and perhaps even reject these proofs as mere heuristic arguments for the protocol's security.

Acknowledgments. We wish to thank Greg Zaverucha for bringing reference [44] to our attention. We also thank Debrup Chakraborty, Koray Karabina, Ann Hibner Koblitz, Neal Koblitz, Berkant Ustaoglu and Greg Zaverucha for commenting on an earlier draft.

References

1. W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr, "RSA and Rabin functions: Certain parts are as hard as the whole", *SIAM J. Computing*, 17 (1988), 194–209.
2. M. Bellare, A. Boldyreva and S. Micali, "Public-key encryption in a multi-user setting: security proofs and improvements", *EUROCRYPT 2000*, LNCS 1807 (2000), 259–274.
3. M. Bellare, R. Canetti and H. Krawczyk, "Keying hash functions for message authentication", *CRYPTO '96*, LNCS 1109 (1996), 1–15.
4. M. Bellare and P. Rogaway, "Entity authentication and key distribution", *CRYPTO '93*, LNCS 775 (1994), 232–249.
5. M. Bellare and P. Rogaway, "The exact security of digital signatures — how to sign with RSA and Rabin", *EUROCRYPT '96*, LNCS 1070 (1996), 399–416.
6. D. Bernstein, "The Poly1305-AES message-authentication code", *FSE 2005*, LNCS 3557 (2005), 32-49.
7. D. Bernstein, "Proving tight security for Rabin-Williams signatures", *EUROCRYPT 2008*, LNCS 4965 (2008), 70–87.
8. R. Bhaskar, J. Herranz and F. Laguillaumie, "Aggregate designated verifier signatures and application to secure routing", *Int. J. Security and Networks*, 2 (2007), 192–201.
9. E. Biham, "How to decrypt or even substitute DES-encrypted messages in 2^{28} steps", *Information Processing Letters*, 84 (2002), 117–124.
10. A. Biryukov, S. Mukhopadhyay and P. Sarkar, "Improved time-memory trade-offs with multiple data", *SAC 2005*, LNCS 3897 (2006), 110–127.
11. A. Biryukov and A. Shamir, "Cryptanalytic time/memory/data tradeoffs for stream ciphers" *ASIACRYPT 2000*, LNCS 1976 (2000), 1–13.
12. J. Black and P. Rogaway, "CBC MACs for arbitrary-length messages: The three-key constructions", *CRYPTO 2000*, LNCS 1880 (2000), 197–215.
13. S. Blake-Wilson, D. Johnson and A. Menezes, "Key agreement protocols and their security analysis", *Cryptography and Coding '97*, LNCS 1355 (1997), 30–45. Full version at http://www.cacr.math.uwaterloo.ca/techreports/1997/corr97-17.ps.
14. L. Blum, M. Blum and M. Shub, "A simple unpredictable pseudo-random number generator", *SIAM J. Computing*, 15 (1986), 364–383.

15. D. Boneh and X. Boyen, "Efficient selective-ID secure identity based encryption schemes without random oracles", *EUROCRYPT 2004*, LNCS 3027 (2004), 223–238.
16. D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing", *SIAM J. Computing*, 32 (2003), 586–615.
17. D. Boneh, C. Gentry, B. Lynn and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps", *EUROCRYPT 2003*, LNCS 2656 (2003), 416–432.
18. X. Boyen, "A tapestry of identity-based encryption: practical frameworks compared", *Int. J. Applied Cryptography*, 1 (2008), 3–21.
19. X. Boyen and L. Martin, "Identity-based cryptography standard (IBCS) #1: Supersingular curve implementations of the BF and BB1 cryptosystems", IETF RFC 5091, 2007.
20. R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels", *EUROCRYPT 2001*, LNCS 2045 (2001), 453–474. Full version at http://eprint.iacr.org/2001/040.
21. H. Chan, A. Perrig and D. Song, "Secure hierarchical in-network aggregation in sensor networks", *CCS '06*, 2006, 278–287.
22. L. Chen and Z. Cheng, "Security proof of Sakai-Kasahara's identity-based encryption scheme", *Cryptography and Coding 2005*, LNCS 3796 (2005), 442–459.
23. J.-S. Coron, "On the exact security of full domain hash", *CRYPTO 2000*, LNCS 1880 (2000), 229–235.
24. J.-S. Coron, "Optimal security proofs for PSS and other signature schemes", *EUROCRYPT 2002*, LNCS 2332 (2002), 272–287.
25. I. Damgård, "A "proof-reading" of some issues in cryptography", *ICALP 2007*, LNCS 4596 (2007), 2–11.
26. Q. Dang, "Recommendation for applications using approved hash algorithms", NIST Special Publication 800-107, 2009.
27. M. Dworkin, "Recommendation for block cipher modes of operation: Methods and techniques", NIST Special Publication 800-38A, 2001.
28. M. Dworkin, "Recommendation for block cipher modes of operation: The CMAC mode for authentication", NIST Special Publication 800-38B, 2005.
29. D. Eastlake, S. Crocker and J. Schiller, "Randomness recommendations for security", IETF RFC 1750, 1994.
30. The eSTREAM project, http://www.ecrypt.eu.org/stream/.
31. O. Eikemeier et al., "History-free aggregate message authentication codes", *SCN 2010*, LNCS 6280 (2010), 309–328.
32. FIPS 180-3, "Secure Hash Standard (SHS)", Federal Information Processing Standards Publication 180-3, National Institute of Standards and Technology, 2008.
33. FIPS 198-1, "The Keyed-Hash Message Authentication Code (HMAC)", Federal Information Processing Standards Publication 198, National Institute of Standards and Technology, 2008.
34. D. Galindo, "Boneh-Franklin identity based encryption revisited", *ICALP 2005*, LNCS 3580 (2005) 791–802.
35. C. Gentry and S. Halevi, "Hierarchical identity based encryption with polynomially many levels", *TCC 2009*, LNCS 5444 (2009), 437–456.
36. O. Goldreich, "On the foundations of modern cryptography", *CRYPTO '97*, LNCS 1294 (1997), 46–74.
37. S. Halevi and P. Rogaway, "A tweakable enciphering mode", *CRYPTO 2003*, LNCS 2729 (2003), 482–499.

38. S. Halevi and P. Rogaway, “A parallelizable enciphering mode”, *CT-RSA 2004*, LNCS 2964 (2004), 292–304.
39. M. Hellman, “A cryptanalytic time-memory trade-off” *IEEE Trans. Info. Th.,* 26 (1980), 401–406.
40. J. Hong and P. Sarkar, “New applications of time memory data tradeoffs”, *ASIACRYPT 2005*, LNCS 3788 (2005), 353–372.
41. M. Jakobsson, K. Sako and R. Impagliazzo, “Designated verifier proofs and their applications”, *EUROCRYPT ’96*, LNCS 1070 (1996), 143–154.
42. S. Kakvi and E. Kiltz, “Optimal security proofs for Full Domain Hash, revisited”, *EUROCRYPT 2012*, LNCS 7237 (2012), 537–553.
43. S. Kakvi, E. Kiltz and A. May, “Certifying RSA”, *ASIACRYPT 2012*, LNCS 7658 (2012), 404–414.
44. J. Katz and A. Lindell, “Aggregate message authentication codes”, *CT-RSA 2008*, LNCS 4964 (2008), 155–169.
45. J. Katz and N. Wang, “Efficiency improvements for signature schemes with tight security reductions”, *CCS ’03*, 2003, 155–164
46. S. Kelly and S. Frankel, “Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec”, IETF RFC 4868, 2007.
47. S. Kent and R. Atkinson, “IP authentication header’, IETF RFC 4302, 2005.
48. N. Koblitz and A. Menezes, “Another look at “provable security””, *J. Cryptology*, 20 (2007), 3–37.
49. N. Koblitz and A. Menezes, “Another look at “provable security”. II”, *Indocrypt 2006*, LNCS 4329 (2006), 148–175.
50. N. Koblitz and A. Menezes, “Another look at HMAC”, Cryptology ePrint Archive: Report 2012/074; available at http://eprint.iacr.org/2012/074.
51. H. Krawczyk, “HMQV: A high-performance secure Diffie-Hellman protocol”, *CRYPTO 2005*, LNCS 3621 (2005), 546–566. Full version at http://eprint.iacr.org/2005/176.
52. K. Kurosawa and T. Iwata, “TMAC: Two-key CBC MAC”, *CT-RSA 2003*, LNCS 2612 (2003), 33–49.
53. B. LaMacchia, K. Lauter and A. Mityagin, “Stronger security of authenticated key exchange”, *ProvSec 2007*, LNCS 4784 (2007), 1–16.
54. S. Lu, R. Ostrovsky, A. Sahai, H. Shacham and B. Waters, “Sequential aggregate signatures and multisignatures without random oracles”, *EUROCRYPT 2006*, LNCS 4004 (2006), 465–485.
55. M. Luby, *Pseudorandomness and Cryptographic Applications*, Princeton University Press, 1996.
56. D. McGrew and J. Viega, “The security and performance of the Galois/Counter Mode (GCM)”, *Indocrypt 2004*, LNCS 3348 (2004), 343–355.
57. A. Menezes and N. Smart, “Security of signature schemes in the multi-user setting”, *Designs, Codes and Cryptography*, 33 (2004), 261–274.
58. A. Menezes and B. Ustaoglu, “Security arguments for the UM key agreement protocol in the NIST SP 800-56A standard”, *ASIACCS ’08*, 2008, 261–270.
59. National Security Agency, “SKIPJACK and KEA algorithm specification”, Version 2.0, May 29 1998.
60. P. Paillier and D. Vergnaud, “Discrete-log-based signatures may not be equivalent to discrete log”, *ASIACRYPT 2005*, LNCS 3788 (2005), 1–20.
61. D. Pointcheval and J. Stern, “Security arguments for digital signatures and blind signatures”, *J. Cryptology*, 13 (2000), 361–396.
62. P. Rogaway, “Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC”, *ASIACRYPT 2004*, LNCS 3329 (2004), 16–31.

63. P. Rogaway, "OCB: Background", http://www.cs.ucdavis.edu/~rogaway/ocb/ocb-faq.htm.
64. P. Rogaway, M. Bellare and J. Black, "OCB: A block-cipher mode of operation for efficient authenticated encryption", *ACM Trans. Information and System Security*, 6 (2003), 365–403.
65. P. Rogaway and T. Shrimpton, "A provable-security treatment of the key-wrap problem", *EUROCRYPT 2006*, LNCS 4004 (2006), 373–390. Full version at http://eprint.iacr.org/2006/221.
66. P. Sarkar, "Pseudo-random functions and parallelizable modes of operations of a block cipher", *IEEE Trans. Info. Th.,* 56 (2010), 4025–4037.
67. P. Sarkar, "Tweakable enciphering schemes using only the encryption function of a block cipher", *Inf. Process. Lett.*, 111 (2011), 945–955.
68. S. Schäge, "Tight proofs for signature schemes without random oracles", *EUROCRYPT 2011*, LNCS 6632 (2011), 189–206.
69. C. Schnorr, "Efficient signature generation for smart cards", *J. Cryptology*, 4 (1991), 161–174.
70. J. Shin, "Enhancing privacy in cryptographic protocols", Ph.D. thesis, University of Maryland, 2009.
71. A. Sidorenko and B. Schoenmakers, "Concrete security of the Blum–Blum–Shub pseudorandom generator", *Cryptography and Coding 2005*, LNCS 3796 (2005), 355–375.
72. JH. Song, R. Poovendran, J. Lee and T. Iwata, "The AES-CMAC algorithm", IETF RFC 4493, 2006.
73. B. Ustaoglu, "Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS", *Designs, Codes and Cryptography*, 46 (2008), 329–342.
74. P. van Oorschot and M. Wiener, "Parallel collision search with cryptanalytic applications", *J. Cryptology*, 12 (1999), 1–28.
75. A. Young and M. Yung, *Malicious Cryptography: Exposing Cryptovirology*, Wiley, 2004.