Simplified Submission of Inputs to Protocols

Douglas Wikström*, `dog@csc.kth.se`

CSC KTH Stockholm, Sweden

Abstract. Consider an electronic election scheme implemented using a mix-net; a large number of voters submit their votes and then a smaller number of servers compute the result. The mix-net accepts an encrypted vote from each voter and outputs the set of votes in sorted order without revealing the permutation used. To ensure a fair election, the votes of corrupt voters should be independent of the votes of honest voters, i.e., some type of non-malleability or plaintext awareness is needed. However, for efficiency reasons the servers typically expect inputs from some homomorphic cryptosystem, which is inherently malleable.

In this paper we consider the problem of how non-malleability can be guaranteed in the submission phase and still allow the servers to start their computation with ciphertexts of the appropriate homomorphic cryptosystem. This can clearly be achieved using general techniques, but we would like a solution which is: (1) provably secure under standard assumptions, (2) non-interactive for the submitting parties, (3) very efficient for all parties in terms of computation and communication.

We give the first solution to this problem which has all these properties. Our solution is surprisingly simple and can be based on various Cramer-Shoup cryptosystems. To capture its security properties we introduce a variation of CCA2-security.

1 Introduction

Mix-Nets. A mix-net is a cryptographic protocol executed by N senders and k mix-servers, where typically N is quite large and k is fairly small, e.g., $N = 10^4$ and $k = 10$. The functionality implemented by a mix-net corresponds to a trusted party that collects inputs from the senders and then outputs the inputs in sorted order. The main application of mix-nets is for electronic elections. All known efficient robust mix-nets exploit the homomorphic properties of cryptosystems such as the El Gamal cryptosystem [13] in an essential way. A problem with using a homomorphic cryptosystem in the submission phase is that corrupted senders can submit inputs that are related to those of honest senders.

Formally, the proof of security fails. When using the simulation paradigm, e.g., universally composable security [4], a mix-net is said to be secure if for every adversary attacking the mix-net there is an ideal adversary (simulator), typically running the adversary as a black-box, attacking the ideal mix-net (the trusted party mentioned above) such that no environment can tell the two models apart. The simulator does not know the inputs of the honest parties and replaces them in its simulation, e.g., by zero messages. It must also extract the inputs of corrupted parties in its simulation and hand them to the ideal mix-net, since otherwise these inputs would be missing in the output from the ideal mix-net and the environment could trivially distinguish the

* Work partly done while at ETH Zürich, Department of Computer Science.

two models. Any successful adversary must result in a successful attack against the underlying cryptosystem, which means that the simulator can not use the secret key of the cryptosystem to extract the inputs of corrupt senders.

General Case. More generally, consider a cryptographic protocol that starts with a submission phase where many parties submit ciphertexts formed with a public key *pk*, and a smaller group of servers hold shares of the secret key *sk* corresponding to *pk*. The servers then compute and publish some function of the input plaintexts. Typically, efficient protocols exploit the algebraic structure of the cryptosystem, e.g., homomorphic properties. The problem with using a polynomially indistinguishable cryptosystem directly in such protocols is that it does not guarantee that the plaintexts submitted by corrupt parties are unrelated to those submitted by honest users.

Formally, the problem surfaces when the cryptographer constructing the protocol tries to reduce the security of his/her scheme to the security of the cryptosystem. If the simulation paradigm is used, some kind of simulator must be constructed and the simulator must extract the values of the corrupt parties to be able to hand these to an ideal version of the protocol. The existence of a successful adversary must contradict the security of the cryptosystem, i.e., extraction must be done without using the secret key of the cryptosystem. This is not possible using only a polynomially indistinguishable [14] cryptosystem.

The Submission Problem. The submission problem is how to find a submission scheme such that: (a) the inputs of corrupted parties can be extracted by the simulator without using the secret key, and (b) its output is a list of ciphertexts of the form expected by the servers computing the result. These requirements are essential to allow use of the submission scheme as a prefix to the main protocol, but there are also several natural additional properties that we can look for, or even require, in the submission phase.

1. The solution should be provably secure under standard assumptions in the plain model, i.e., without any random oracles or generic groups.
2. The submission of an input should be non-interactive for a submitting party and interaction between the servers should be limited, e.g., independent of the number of senders.
3. The communication and computational complexity of the submittors should not depend on the number of servers.

1.1 Previous Work

Informally, we may view any solution to the problem as a form of proof of knowledge of the encrypted plaintext, since any solution must allow the simulator to extract the submitted plaintext without knowledge of the secret key of the polynomially indistinguishable cryptosystem. We classify the solutions in the literature and some minor extensions as follows:

1. An *interactive* proof of knowledge [15] is used, either with a straight-line extractor in the public key setting using the Naor and Yung [18] double-ciphertext trick, or with a rewinding extractor.

2. A non-interactive proof of knowledge using *general techniques* [2] is used. This is *not efficient* for either the submittor or the servers, even using the recent techniques of Groth et al. [16]. Note that this is essentially the construction of a CCA2-secure cryptosystems under general assumptions given by Sahai [23] based on the Naor-Yung double-ciphertext trick, but starting from a concrete polynomially indistinguishable cryptosystem with useful structure.
3. A non-interactive (for the submitter) proof of knowledge based on verifiable secret sharing is used, for example using techniques from Abe, Cramer, and Fehr [1]. Then the *computational and communication complexity of the submitting party grows linearly* with the number of servers, since each server must receive an encrypted secret share, and the servers must communicate for each submitted ciphertext to verify its proof.
4. A non-interactive proof of knowledge in the *random oracle model* is used, either using the Naor and Yung double ciphertext trick, or with rewinding. Such solutions are typically efficient, but unfortunately only heuristically secure [5]. Note that the CCA2-secure cryptosystems in the random oracle model given by Shoup and Gennaro [25] may be viewed as instantiations of this solution.
5. An arbitrary CCA2-secure cryptosystem is used and ciphertexts are translated into suitable polynomially indistinguishable ciphertexts using *general multiparty computation* techniques. This is inefficient both in terms of computation and communication.
6. A special CCA2-secure cryptosystem such that a ciphertext can be transformed more easily into a new ciphertext for the basic polynomially indistinguishable scheme needed by the servers is used. We list the solutions of this type we are aware of below.
   (a) Canetti and Goldwasser [6] and Lysyanskaya and Peikert [17] have given CCA2-secure cryptosystems with distributed decryption which allows transforming ciphertexts into ciphertexts for polynomially indistinguishable cryptosystems. These either involve *interaction*, *expensive setup assumptions*, or only work for a *small number of servers.*
   (b) Boneh, Boyen, and Halevi [3] give a CCA2-secure cryptosystem with distributed decryption that may be viewed as containing a polynomially indistinguishable cryptosystem, but its security is based on a *non-standard complexity assumption* based on pairings.
   (c) Cramer, Damgård, and Ishai [7] present a solution based on distributed pseudo random functions and share conversion that is reasonably efficient for a *small number of servers.*

To summarize, there are numerous solutions to the submission problem which satisfies properties (a) and (b), but no such solution has the properties (1)-(3) listed above.

1.2 Our Contribution

We give a simple solution to the submission problem that is efficient both in terms of computation and communication.

The Idea. Recall the original construction of Cramer and Shoup [9]. The cryptosystem is deployed in a group G_q of prime order in which the decision Diffie-Hellman assumption is assumed to be hard. The key generator first chooses two random generators $g_0, g_1 \in G_q$. Then it chooses random exponents $x_0, x_1, y_0, y_1, z \in \mathbb{Z}_q$ and defines $c = g_0^{x_0} g_1^{x_1}$, $d = g_0^{y_0} g_1^{y_1}$, and $h = g_0^z$. It also generates a collision-free hash function H. Finally, it outputs $(pk, sk) = ((H, g_0, g_1, c, d, h), (x_0, x_1, y_0, y_1, z))$. To encrypt a message $m \in G_q$ using the public key pk the encryption algorithm chooses $r \in \mathbb{Z}_q$ randomly and outputs the tuple $(u_0, u_1, e, v) = (g_0^r, g_1^r, h^r m, c^r d^{rH(u_0, u_1, e)})$. To decrypt a tuple $(u_0, u_1, e, v) \in G_q^4$ using the secret key sk the decryption algorithm first checks the validity of the ciphertext by testing if $u_0^{x_0} u_1^{x_1} (u_0^{y_0} u_1^{y_1})^{H(u_0, u_1, e)} = v$. If so it outputs e/u_0^z, and otherwise the unit element, i.e., 1, of the group.[1]

Note that $h = g^z$ and z have the form of an El Gamal [13] public and secret key respectively and that (u_0, e) is nothing more than an El Gamal ciphertext. This is of course not a new observation. What seems to be a new observation is the fact that the holder of the secret key may reveal (x_0, x_1, y_0, y_1) without any loss in security as long as it never decrypts any ciphertext constructed after this point, and that this solves the submission problem.

Generalizing and Applying the Idea. To allow us to generalize the observation about the original Cramer-Shoup scheme and identify a class of cryptosystems for which it applies, we introduce the notion of an *augmented cryptosystem* which contains another cryptosystem as a component. In applications the latter cryptosystem will have some useful structure, e.g., be homomorphic, that allows more efficient and simpler protocols. We also introduce a strengthened variation of CCA2-security called *submission security* and observe that the generic scheme of Cramer and Shoup [10] already satisfies this stronger definition. Finally, we illustrate the use of the new notion by applying it to general secure function evaluation, which strictly generalizes the notion of a mix-net.

Limitations of Our Approach. When using our solution, no new inputs can be accepted after part of the secret key is revealed. This is a minor drawback in the targeted applications, since we have a large number of submitting parties and executions of the underlying protocol are infrequent. When a new session is executed the servers simply generate a new key. However, it may be useful to re-use the public key of the basic cryptosystem in the underlying protocol. Thus, our definitions require that the augmentation can be replaced by a freshly generated augmentation without any loss in security. This allows using several independent augmentations that may be revealed sequentially, i.e., inputs can be processed in batches and then input to the same underlying protocol. We remark that for general threshold decryption, e.g. [6], our approach is not reasonable, since users requesting a decryption expect the result immediately.

[1] In [9] a special symbol $\perp$ is output if the test fails, but this is only to simplify the analysis. Any fixed output works just as well.

1.3 Notation

We denote by PT and PPT the sets of deterministic and probabilistic polynomial time Turing machines respectively, and write PT* for the set of non-uniform polynomial time Turing machines. We use n to denote the security parameter, and say that a function $\epsilon(n)$ is negligible if for every constant c there exists a constant n_0 such that $\epsilon(n) < n^{-c}$ for $n > n_0$. If pk is the public key of a cryptosystem, we denote by $\mathcal{M}_{pk}$, $\mathcal{C}_{pk}$, and $\mathcal{R}_{pk}$ the plaintext space, the ciphertext space, and the randomness space respectively.

2 Augmented Cryptosystems

Keeping our observation about the original Cramer-Shoup cryptosystem in mind, we formalize a general class of *augmented* cryptosystems that given part of the secret key allow conversion of a ciphertext into a ciphertext of another *basic* cryptosystem. In applications, the basic cryptosystem typically has special properties, e.g., it is homomorphic, that are exploited by the cryptographic protocol. We introduce the following definition.

Definition 1 (Augmented Cryptosystem). *A cryptosystem* $\mathcal{CS} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ *is an augmentation of a cryptosystem* $\mathcal{CS}^B = (\mathsf{Kg}^B, \mathsf{Enc}^B, \mathsf{Dec}^B)$ *if there exists an* augmentation algorithm $\mathsf{Aug} \in$ PPT *and a* stripping algorithm $\mathsf{Strip} \in$ PT *such that:*

1. *On input* 1^n, Kg *computes* $(pk^B, sk^B) = \mathsf{Kg}^B(1^n)$ *and* $(pk^A, sk^A) = \mathsf{Aug}(pk^B)$, *and outputs* $(pk, sk) = ((pk^A : pk^B), (sk^A : sk^B))$.
2. *On input* $((sk^A : sk^B), c)$, Dec *outputs* $\mathsf{Dec}^B_{sk^B}(\mathsf{Strip}_{pk,sk^A}(c))$.

Clearly, any cryptosystem can be viewed as a trivial augmentation of itself, and if it is CCA2-secure then the trivial augmentation is also submission secure as defined below. We are interested in non-trivial augmentations where $\mathcal{CS}^B$ has structural properties useful in the construction of protocols.

Some readers may find it tempting to use a definition that mirrors the Cramer-Shoup cryptosystem more closely to avoid the existence of trivial augmentations, i.e., one could explicitly require that it is possible to check the "validity" of a ciphertext using sk^A. We remind those readers that for most cryptographic notions there are trivial instances, e.g., the identity map is a cryptosystem, and we see no reason to impose unnecessary conditions on which particular properties of the basic cryptosystem that should be considered useful.

2.1 Submission Security of Augmented Cryptosystems

Recall the game considered in the definition of CCA2-security [18,11,22]. The adversary is given a public key pk. Then it may ask any number of decryption queries to a decryption oracle $\mathsf{Dec}_{sk}(\cdot)$ holding the secret key sk corresponding to pk. The adversary must then choose two challenge messages m_0 and m_1. The game is parameterized by a bit b and returns a challenge ciphertext of the form $c = \mathsf{Enc}_{pk}(m_b)$. Then the adversary is again allowed to ask arbitrary decryption queries to $\mathsf{Dec}_{sk}(\cdot)$ with the exception of c, and must finally output a guess b' of the bit b. If the cryptosystem is CCA2-secure, then the difference in distribution of b' when $b = 0$ or $b = 1$ respectively, should be negligible.

In the game we consider, the adversary is given a public key pk^B of the basic cryptosystem. It can request that the experiment generates a fresh augmentation $(pk_j^A, sk_j^A) = \mathsf{Aug}(pk^B)$, stores $(pk_j, sk_j) = ((pk_j^A : pk^B), (sk_j^A : sk^B))$, and returns $pk_j = (pk_j^A : pk^B)$ to the adversary. This is done by submitting the integer j to its $pk_{(\cdot)}^A$-oracle. Any subsequent identical queries j give the same pk_j. It can ask decryption queries. This is done by submitting an index and ciphertext pair (j, c') to its $\mathsf{Dec}_{sk_{(\cdot)}}(\cdot)$-oracle. It can request that the experiment reveals an augmentation sk_j^A by submitting j to its $sk_{(\cdot)}^A$-oracle, but after such a query no more decryption queries of the form (j, c') for some ciphertext c' are allowed. Then the adversary must choose an index i and two challenge messages m_0 and m_1. The game is parameterized by a bit b and returns a challenge ciphertext of the form $c = \mathsf{Enc}_{pk_i}(m_b)$. The adversary is then again allowed to: ask for more fresh public keys, ask more decryption queries with the exception of decryption of (i, c), and request more augmentations. Finally, it outputs a guess b' of b. If the cryptosystem is submission secure, then the difference in distributions of b' with $b = 0$ or $b = 1$ respectively should be negligible.

Experiment 1 (Submission Security, $\mathsf{Exp}^{\mathsf{sub}-b}_{\mathcal{CS},\mathcal{CS}^B,A}(n)$).

$$
\begin{aligned}
(pk^B, sk^B) &\leftarrow \mathsf{Kg}^B(1^n) && \textit{// Basic keys}\\
(pk_j^A, sk_j^A) &\leftarrow \mathsf{Aug}(pk^B) \quad \textit{for } j = 1, 2, 3, \ldots && \textit{// Augmentations}\\
(pk_j, sk_j) &\leftarrow \big((pk_j^A : pk^B), (sk_j^A : sk^B)\big) && \textit{// Augmented keys}\\
(i, m_0, m_1, state) &\leftarrow A^{pk_{(\cdot)}^A, sk_{(\cdot)}^A, \mathsf{Dec}_{sk_{(\cdot)}}(\cdot)}(\mathsf{choose}, pk^B) && \textit{// Choice of challenges}\\
c &\leftarrow \mathsf{Enc}_{pk_i}(m_b) && \textit{// Challenge ciphertext}\\
d &\leftarrow A^{pk_{(\cdot)}^A, sk_{(\cdot)}^A, \mathsf{Dec}_{sk_{(\cdot)}}(\cdot)}(\mathsf{guess}, state) && \textit{// Guess of adversary}
\end{aligned}
$$

The experiment returns 0 if $\mathsf{Dec}_{sk_{(\cdot)}}(\cdot)$ *was queried on* (i, c) *or if it was queried on* (j, c') *for some* c' *after* $sk_{(\cdot)}^A$ *was queried on* j*. Otherwise the experiment returns* d*.*

We could equivalently have defined a game where the game only generates an augmentation if requested to do so by the adversary, but the above is conceptually simpler.

Definition 2 (Submission Security). *An augmentation* $\mathcal{CS}$ *of* $\mathcal{CS}^B$ *is said to be submission secure if* $\forall A \in \mathrm{PT}^*$*:* $|\Pr[\mathsf{Exp}^{\mathsf{sub}-0}_{\mathcal{CS},\mathcal{CS}^B A}(n) = 1] - \Pr[\mathsf{Exp}^{\mathsf{sub}-1}_{\mathcal{CS},\mathcal{CS}^B,A}(n) = 1]|$ *is negligible.*

Example 1. A simple example of a submission secure cryptosystem can be derived from the scheme of Sahai [23] based on the Naor and Yung double ciphertext trick [18]. A polynomially indistinguishable cryptosystem $\mathcal{CS}^B$ is given and a CCA2-secure cryptosystem $\mathcal{CS} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ is constructed as follows. To generate a public key, compute $(pk_0^B, sk_0^B) = \mathsf{Kg}^B(1^n)$ and $(pk_1^B, sk_1^B) = \mathsf{Kg}^B(1^n)$, and generate a common reference string CRS. Then output $(pk, sk) = ((pk_0^B, pk_1^B, CRS), (sk_0^B, sk_1^B))$. To encrypt a message m, output $(c_0, c_1, \pi) = (\mathsf{Enc}^B_{pk_0^B}(m), \mathsf{Enc}^B_{pk_1^B}(m), \pi)$, where π is a simulation sound non-interactive adaptively zero-knowledge proof (NIZKP) that the same message is encrypted in c_0 and c_1. To decrypt, verify the NIZKP and output $\mathsf{Dec}^B_{sk_0}(c_0)$ or 0 depending on if the NIZKP was accepted or not.

We could define a cryptosystem $\mathcal{CS}' = (\mathsf{Kg}, \mathsf{Enc}^B, \mathsf{Dec}^B)$ where the latter two algorithms are executed on pk_0^B and sk_0^B and ignore the remainder of the keys. The augmentation algorithm Aug could give an empty pair as output, and we could define the stripping algorithm Strip as checking the NIZKP and outputting c_0 or $\mathsf{Enc}_{pk_0^B}(0, 0)$ depending on if the NIZKP was accepted or not.

If we view $\mathcal{CS}$ as an augmentation of $\mathcal{CS}'$ it is submission secure. Although this example does not satisfy our additional requirements of a solution to the submission problem (it is very inefficient) it does simplify the analysis of the submission phase of a theoretical construction based on such a cryptosystem.

3 Generic Cramer-Shoup Is Submission Secure

The fact that the generic CCA2-secure cryptosystem of Cramer and Shoup is submission secure if we view it as an augmentation of a basic polynomially indistinguishable cryptosystem is quite easy to see from their security proof. On the other hand we need to show that this is indeed the case. Thus, we recall their scheme and prove this fact in the appendix, but we use coarse-grained and streamlined definitions. We also take the liberty of ignoring the technical problem of constructing efficiently computable hash families, since this complicates the definitions and does not add anything to our exposition (see [10] for details).

3.1 Preliminaries

Subset Membership Problems. A subset membership problem consists of three sets X, $L \subsetneq X$, and W, and a relation $R \subset X \times W$. The idea is that it should be hard to decide if an element is sampled from L or from $X \setminus L$. To be useful in cryptography we also need some algorithms that allow us to sample instances and elements, and check for membership in X.

Definition 3. *A subset membership problem $\mathbb{M}$ consists of a collection of distributions $(I_n)_{n\in\mathbb{N}}$, an instance generator* $\mathsf{Ins} \in \mathrm{PPT}$*, a sampling algorithm* $\mathsf{Sam} \in \mathrm{PPT}$*, and a membership checking algorithm* $\mathsf{Mem} \in \mathrm{PT}$ *such that:*

1. *I_n is a distribution on instance descriptions $\Lambda[X, L, W, R]$ specifying finite non-empty sets X, $L \subsetneq X$, and W, and a binary relation $R \subset X \times W$.*
2. *On input 1^n,* Ins *outputs an instance Λ with distribution statistically close to I_n.*
3. *On input 1^n and $\Lambda[X, L, W, R] \in [I_n]$,* Sam *outputs $(x, w) \in R$, where the distribution of x is statistically close to uniform over L.*

4. *On input 1^n, $\Lambda[X,L,W,R] \in [I_n]$, and $\zeta \in \{0,1\}^*$,* Mem *outputs* 1 *or* 0 *depending on if $\zeta \in X$ or not.*

Definition 4. *Let $\mathbb{M}$ be a subset membership problem. Sample Λ from I_n and let x and x' be randomly distributed over L and $X \setminus L$ respectively. We say that $\mathbb{M}$ is hard if for all $A \in \mathrm{PT}^*$: $|\Pr[A(\Lambda, x) = 1] - \Pr[A(\Lambda, x') = 1]|$ is negligible.*

Hash Families. Hash families are well known in the cryptographic literature and there are many variations. We assume that all families are indexed by a security parameter n.

Definition 5. *A projective hash family $\mathbb{H} = (H, K, X, L, \Pi, S, \alpha)$ consists of finite non-empty sets K, X, $L \subsetneq X$, Π, and S, a function $\alpha : K \to S$, and a collection of hash functions $H = (H_k : X \times \Pi \to \Pi)_{k \in K}$, where $\alpha(k)$ determines H_k on $L \times \Pi$.*

Definition 6. *Let $\mathbb{H} = (H, K, X, L, \Pi, S, \alpha)$ be a projective hash family, and let $k \in K$ be random. Then $\mathbb{H}$ is universal$_2$ if for all $s \in S$, $x, x' \in X$ with $x \notin L \cup \{x'\}$, and $\pi_x, \pi'_x, \pi, \pi' \in \Pi$, $\Pr_k[H_k(x, \pi_x) = \pi \wedge H_k(x', \pi'_x) = \pi' \wedge \alpha(k) = s]$ is negligible.*

The following definition and lemma are stated informally in [10].

Experiment 2 (Computationally Universal$_2$, $\mathsf{Exp}^{\mathsf{cuni}_2}_{\mathbb{H},A}(n)$). *Let τ_k be the predicate defined by $\tau_k((x, \pi_x), \pi) \iff H_k(x, \pi_x) = \pi$, and consider the following experiment.*

$$\begin{aligned} k &\leftarrow K \\ (x, \pi_x, state) &\leftarrow A^{\tau_k(\cdot,\cdot)}(\alpha(k)) \\ &\leftarrow A^{\tau_k(\cdot,\cdot)}(H_k(x, \pi_x), state) \end{aligned}$$

Denote by $((x_i, \pi_{x,i}), \pi_i)$ the ith query to τ_k, and let i_l be the index of the last query before the first output. If A asks a query $((x_i, \pi_{x,i}), \pi_i)$ to τ_k with $H_k(x_i, \pi_{x,i}) = \pi_i$, $x_i \in X \setminus L$, and $i \le i_l$ or $x_i \neq x$, then output 1 and otherwise 0.

Definition 7. *A projective hash family $\mathbb{H}$ is computationally universal$_2$ if for every $A \in \mathrm{PT}^*$, $\Pr[\mathsf{Exp}^{\mathsf{cuni}_2}_{\mathbb{H},A}(n) = 1]$ is negligible.*

Lemma 1. *If a projective hash family $\mathbb{H}$ is universal$_2$, then it is computationally universal$_2$.*

Definition 8. *Let $\mathbb{H} = (H, K, X, L, \Pi, S, \alpha)$ be a projective hash family, and let $k \in K$, $x \in X \setminus L$, and $\pi \in X$ be random. Then $\mathbb{H}$ is smooth if for every $\pi_x \in \Pi$ the distributions of $(x, \pi_x, \alpha(k), H_k(x, \pi_x))$ and $(x, \pi_x, \alpha(k), \pi)$ are statistically close.*

Universal Hash Proof Systems. Informally, a hash proof system may be viewed as a non-interactive zero-knowledge proof system where only the holder of a secret key corresponding to the public common random string can verify a proof. Strictly speaking, the definition below corresponds, in the unconditional case, to a special case of what Cramer and Shoup [10] call "extended strongly (smooth, universal$_2$)" hash proof system.

Definition 9. *A (smooth, (computational) universal$_2$) hash proof system $\mathbb{P}$ for a subset membership problem $\mathbb{M}$ associates with each instance $\Lambda[X, L, W, R]$ a (smooth, (computationally) universal$_2$) projective hash family $\mathbb{H} = (H, K, X, L, \Pi, S, \alpha)$, and the following algorithms*

1. *A key generation algorithm* $\mathsf{Gen} \in \mathrm{PPT}$ *that on input 1^n and $\Lambda \in [I_n]$ outputs (s, k), where k is randomly distributed in K and $s = \alpha(k)$.*
2. *A private evaluation algorithm* $\mathsf{PEval} \in \mathrm{PT}$ *that on input 1^n, $\Lambda \in [I_n]$, $k \in K$, and $(x, \pi_x) \in X \times \Pi$ outputs $H_k(x, \pi_x)$.*
3. *A public evaluation algorithm* $\mathsf{Eval} \in \mathrm{PT}$ *that on input 1^n, $\Lambda \in [I_n]$, $\alpha(k)$ with $k \in K$, (x, π_x) and w, with $(x, w) \in R$ and $\pi_x \in \Pi$, outputs $H_k(x, \pi_x)$.*
4. *A membership checking algorithm* $\mathsf{Mem} \in \mathrm{PT}$ *that on input 1^n, $\Lambda \in [I_n]$, and $\zeta \in \{0,1\}^*$ outputs 1 or 0 depending on if $\zeta \in \Pi$ or not.*

3.2 Generic Scheme of Cramer and Shoup

Given the definitions above it is not hard to describe the generic cryptosystem of Cramer and Shoup [10]. Let $\mathbb{M}$ be a hard subset membership problem, such that Π can be fitted with a group operation for any Λ, let $\mathbb{P}_0 = (\mathsf{Gen}_0, \mathsf{PEval}_0, \mathsf{Eval}_0, \mathsf{Mem}_0)$ be a smooth hash proof system for $\mathbb{M}$, and let $\mathbb{P}_1 = (\mathsf{Gen}_1, \mathsf{PEval}_1, \mathsf{Eval}_1, \mathsf{Mem}_1)$ be a computationally universal$_2$ hash proof system for $\mathbb{M}$.

Key Generation. Compute $\Lambda[X, L, W, R] = \mathsf{Ins}(1^n)$, $(s, k) = \mathsf{Gen}_0(1^n, \Lambda)$, $(\widehat{s}, \widehat{k}) = \mathsf{Gen}_1(1^n, \Lambda)$, and output the key pair $(pk, sk) = ((\widehat{s} : \Lambda, s), (\widehat{k} : k))$.
Encryption of a message $m \in \Pi$. Compute $(x, w) = \mathsf{Sam}(\Lambda)$, $\pi = \mathsf{Eval}_0(\Lambda, s, x, w) = H_k(x)$, $e = m + \pi$, and $\widehat{\pi} = \mathsf{Eval}_1(\Lambda, \widehat{s}, x, w, e) = \widehat{H}_{\widehat{k}}(x, e)$, and output $(x, e, \widehat{\pi})$.
Decryption of a ciphertext $(x, e, \widehat{\pi})$. Output $m = e - \mathsf{PEval}_0(\Lambda, k, x) = e - H_k(x)$, only if $\mathsf{PEval}_1(\Lambda, \widehat{k}, x, e) = \widehat{H}_{\widehat{k}}(x, e) = \widehat{\pi}$ and otherwise output 0.

We have not modified the cryptosystem, except that we have introduced a comma in the notation to distinguish the two parts of the public and secret keys as needed in the definition of submission security, and we have replaced the special symbol $\perp$ by the zero plaintext.

Write $\mathcal{CS} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ for the cryptosystem, and let $\mathcal{CS}^B = (\mathsf{Kg}^B, \mathsf{Enc}^B, \mathsf{Dec}^B)$ be the underlying basic cryptosystem defined as follows. It has public key (Λ, s) and secret key k. A message $m \in \Pi$ is encrypted as (x, e), where $e = \mathsf{Eval}_0(\Lambda, s, x, w) + m$, and a ciphertext (x, e) is decrypted by computing $e - \mathsf{PEval}_0(\Lambda, k, x)$. It is clear that $\mathcal{CS}$ is an augmentation of $\mathcal{CS}^B$, since we can define $\mathsf{Aug}(\Lambda, s) = \mathsf{Gen}_1(1^n, \Lambda)$ and define $\mathsf{Strip}_{pk, \widehat{k}}(x, e, \widehat{\pi})$ to output (x, e) if $\mathsf{PEval}_1(\Lambda, \widehat{k}, x, e) = \widehat{\pi}$ and otherwise $\mathsf{Enc}_{pk^B}(0, 0)$.

Cramer and Shoup prove (see Theorem 1 in [10]) that $\mathcal{CS}$ is CCA2-secure under the assumption that $\mathbb{M}$ is hard. We show a stronger result.

Proposition 1. *If $\mathbb{M}$ is a hard subset membership problem, then $\mathcal{CS}$ is a submission secure augmentation of $\mathcal{CS}^B$.*

For a proof of the proposition we refer the reader to the appendix. The key observations needed to extend Cramer's and Shoup's proof of CCA2-security are:

1. The projective property of hash proofs implies that proofs computed using a witness and hash proofs computed using the private key $\widehat{k}$ are *identical* (indeed this is how a hash proof is verified). This means that the holder of $\widehat{k}$ can "perfectly simulate" proofs without the witness, i.e., *even if* $\widehat{k}$ *is made public* the "simulated proof" looks *exactly* like a proof computed using a witness.
2. The soundness of proofs computed by an adversary *before* $\widehat{k}$ is made public, is not decreased by the publishing of $\widehat{k}$.

The generic Cramer-Shoup scheme generalizes several concrete schemes described in [10], such as the El Gamal based scheme in the introduction, but also schemes based on the Paillier cryposystem [19]. Both schemes are common in efficient protocols.

4 Applications of Submission Security

The original motivation for this paper was to come up with a practical non-interactive submission phase in El Gamal based mix-nets. For readers that are not familiar with mix-nets we give an informal description of a construction that goes back to Sako and Kilian [24].

Then we illustrate how the notion of submission secure augmented cryptosystems can be used to construct and analyze the submission phase of a protocol in a modularized way for general secure function evaluation, and explain how this generalizes the mix-net setting in the informal description.

4.1 Informal Description of Application to a Mix-Net

There are many senders $S_1, \ldots, S_N$ and a small number of mix-servers $M_1, \ldots, M_k$, e.g., $N = 10^4$ and $k = 10$. In a joint key generation phase the mix-servers generate a joint public key (g, h) such that each mix-server holds a verifiable secret share s_j of the joint secret key z such that $h = g^z$. To submit a message $m_i \in G_q$ a sender S_i computes an El Gamal ciphertext $(u_{0,i}, e_{0,i}) = (g^{r_i}, h^{r_i} m_i)$, where $r_i \in \mathbb{Z}_q$ is randomly chosen. Then the mix-servers take turns at re-encrypting, using the homomorphic property of El Gamal, and permuting the list of ciphertexts. In other words, for $j = 1, \ldots, k$, M_j computes and publishes $\{(u_{j,i}, e_{j,i})\} = \{(u_{j-1,\pi_j(i)} g^{t_{j,i}}, e_{j-1,\pi_j(i)} h^{t_{j,i}})\}$, where $t_{j,i} \in \mathbb{Z}_q$ and π_j are random. Finally, the mix-servers jointly and verifiably decrypt the list $\{(u_{k,i}, e_{k,i})\}$ output by the last mix-server M_k using their shares s_j, sort the result, and output it.

The idea is that due to the transformations computed by the mix-servers the correspondence between the output plaintexts and the input ciphertexts should be hidden. To ensure robustness, each mix-server also prove correctness of the transformation it computes on the list of ciphertexts (in fact it must prove *knowledge* of a witness of the transformation as pointed out in [27]).

Unfortunately, the construction is completely insecure [21], since a malicious sender S_l may compute its ciphertext as $(u_{0,l}, e_{0,l}) = (u_{0,i}^a, e_{0,i}^a)$ for some random exponent a and then identify a matching pair (m, m^a) in the final output. This reveals the message sent by the honest sender S_i. Intuitively, what is needed is a non-malleable cryptosystem, but on the other hand the cryptosystem must be homomorphic for re-encryption to be possible. Formally, what is needed in the overall proof of security of the mix-net (see [26,27,29]) is a way to extract the messages submitted by corrupted players without using the secret key of the cryptosystem, as explained in the introduction. In previous work this is either solved heuristically, or as in the cited works a proof of knowledge is used explicitly.

We augment the above to make the cryptosystem used for submission identical to the Cramer-Shoup scheme. We set $g_0 = g$ and let the mix-servers generate $g_1 \in G_q$, $x_0, x_1, y_0, y_1 \in \mathbb{Z}_q$, $c = g_0^{x_0} g_1^{x_1}$, and $d = g_0^{y_0} g_1^{y_1}$, where x_0, x_1, y_0, y_1 are verifiably secret shared among the mix-servers. This gives a Cramer-Shoup key pair $((H, g_1, c, d : g_0, h), (x_0, x_1, y_0, y_1 : z))$ with verifiably secret shared secret key. Due to the submission security of the cryptosystem the mix-servers may simply reconstruct the first part (x_0, x_1, y_0, y_1) of the shared key before starting the mixing process. This allows each mix-server to identify the valid ciphertexts *without any additional communication*, and form the list of El Gamal ciphertexts consisting of the El Gamal part of each valid ciphertext. Then the mix-servers execute as explained above.

4.2 Application to a General Functionality

In this section we give a proof of concept for augmented cryptosystem and submission security. We do this by showing how the submission phase can be factored out in the construction and analysis of a multiparty protocol using a submission secure augmented cryptosystem. To keep the exposition clear and ignore irrelevant problems, we show this in the UC-framework [4] and assume the existence of ideal functionalities for the building blocks of the protocol we realize. For simplicity we assume that the adversary can not influence if or when messages are delivered.

Ideal Secure Function Evaluation. Below we formalize secure function evaluation as an ideal functionality. We focus on situations where a large group of submittors give inputs and a smaller group of servers compute the result.

Functionality 1 (Secure Function Evaluation). The function evaluator $\mathcal{F}_f$ for a function f running with submittors $S_1, \ldots, S_N$, servers $M_1, \ldots, M_k$, and ideal adversary $\mathcal{S}$ proceeds as follows.

1. Initialize $D = \emptyset$, $D_S = \emptyset$, and $D_M = \emptyset$, and repeatedly wait for inputs:
   – On input $(\texttt{Input}, x_i)$ from S_i with $i \notin D_S$, set $D \leftarrow D \cup \{(i, x_i)\}$ and $D_S \leftarrow D_S \cup \{i\}$, and then hand $(\texttt{Input}, i)$ to $\mathcal{S}$.
   – On input $(\texttt{RequestOutput})$ from M_j with $j \notin D_M$, set $D_M \leftarrow D_M \cup \{j\}$ and hand $(\texttt{RequestOutput}, j)$ to $\mathcal{S}$. If $|D_M| \geq (k+1)/2$, then go to Step 2.
2. Set $x_i = 0$ for $i \notin D_S$ and hand $(\texttt{Output}, f(x_1, \ldots, x_N))$ to $\mathcal{S}$ and all M_j.

The functionality simply accepts inputs from the submittor (at most one input for each submittor), waits until a majority of the servers request the result, and computes and outputs the result. Recall that the ideal mix-net functionality [26] accepts inputs from senders until a majority of the mix-servers request the sorted list of inputs from senders. Thus, this is an instance of this functionality.

Ideal Secure Function Evaluation On Encrypted Inputs. Consider now the functionality below, which is parameterized by a function f and a cryptosystem $\mathcal{CS}$.

Functionality 2 (Secure Function Evaluation On Encrypted Inputs). The function evaluator $\mathcal{F}_f^{\mathcal{CS}}$ for a function f with cryptosystem $\mathcal{CS}$, running with parties $M_1, \ldots, M_k$, and ideal adversary $\mathcal{S}$ proceeds as follows.

1. Generate a key pair $(pk, sk) = \mathsf{Kg}(1^n)$, and hand $(\texttt{PublicKey}, pk)$ to $\mathcal{S}$ and all M_j.
2. Initialize $D = \emptyset$ and $D_M = \emptyset$, and repeatedly wait for inputs:
   - On input $(\texttt{Input}, C_j)$ from M_j such that $j \notin D_M$, set $D \leftarrow D \cup \{(j, C_j)\}$ and $D_M \leftarrow D_M \cup \{j\}$, and hand $(\texttt{Input}, C_j)$ to $\mathcal{S}$. If $|\{(j, C_j) : C_j = C\}| \geq (k+1)/2$ for some C, then go to Step 3.
3. Interpret C as a list $(c_1, \ldots, c_l)$ of ciphertexts, compute $x = (\mathsf{Dec}_{sk}(c_1), \ldots, \mathsf{Dec}_{sk}(c_l))$ and $y = f(x)$, and hand $(\texttt{Output}, y)$ to $\mathcal{S}$ and all M_j.

The functionality generates and outputs a public key pk of the cryptosystem. Then it waits for inputs, i.e., lists of ciphertexts, until a majority of the servers have submitted identical lists. Finally, it decrypts these ciphertexts, invokes the function f on the resulting plaintexts, and outputs the result.

An ideal functionality of the mixing phase of the mix-net described above which takes El Gamal ciphertexts as input could be modeled by simply letting f be the function that lexicographically sorts its inputs. The first key generation step then corresponds to generating a joint El Gamal public key h with verifiably shared secret key z. This can be done by letting each party generate an El Gamal key pair $((g, h_j), z_j)$ and run Feldman's verifiable secret sharing protocol [12] using the secret key z_j as secret and the public key h_j as the first checking element. Then the joint public key h is the product $\prod h_j$ of all the individual public keys (keys with incorrect sharings are ignored).[2]

Ideal Augmentation. We use the following ideal key augmentation functionality.

Functionality 3 (Augmentation). The cryptosystem augmentor $\mathcal{F}_{aug}^{\mathcal{CS},\mathcal{CS}^B}$ for an augmentation $\mathcal{CS}$ of $\mathcal{CS}^B$ running with servers $M_1, \ldots, M_k$, and ideal adversary $\mathcal{S}$ proceeds as follows.

1. Initialize $D = \emptyset$ and $D_M = \emptyset$ and repeatedly wait for inputs:
   - On input $(\texttt{Augment}, pk_j^B)$ from M_j such that $j \notin D_M$, set $D \leftarrow D \cup \{(j, pk_j^B)\}$ and $D_M \leftarrow D_M \cup \{j\}$. If $|\{(j, pk_j^B) : pk_j^B = pk^B\}| \geq (k+1)/2$ for some pk^B, go to Step 2.
2. Compute $(pk^A, sk^A) = \mathsf{Aug}(pk^B)$, set $pk = (pk^A : pk^B)$ and output $(\texttt{PublicKey}, pk)$.
3. Set $D_M \leftarrow \emptyset$, and and repeatedly wait for inputs:
   - On input $(\texttt{Recover})$ from M_j such that $j \notin D_M$, set $D_M \leftarrow D_M \cup \{j\}$.
     If $|D_M| \geq (k+1)/2$, then output $(\texttt{PrivateKeyAugmentation}, sk^A)$.

The functionality waits until a majority of the servers input the same public key pk^B from the cryptosystem $\mathcal{CS}^B$. It then computes an augmentation $(pk^A, sk^A) = \mathsf{Aug}(pk^B)$ of pk^B (and the unknown sk^B), and outputs the resulting public key $pk = (pk^A : pk^B)$. If later a majority of the mix-servers request sk^A it outputs it.

[2] The adversary can bias the distribution of the resulting joint public key, but only in a benign way. To keep Functionality 2 simple and let us focus on our contribution we ignore this issue.

Note that this corresponds to extending the generation of the joint El Gamal public key to the generation of a joint Cramer-Shoup key. The joint "independent" generator g_1 can be generated exactly as the key h. The joint verification element c can be generated by letting each server M_j generate its own $(x_{j,0}, x_{j,1})$ and c_j and invoking Pedersen verifiable secret sharing [20] where the secret shared is $x_{j,0}$, the constant component of the usually completely random polynomial is fixed to $x_{j,1}$, and c_j is the first checking element. Then the joint element c is defined as the product $\prod c_j$ of all the c_j (keys with incorrect sharings are ignored). The joint element d is generated correspondingly.[3] The recovery phase corresponds to the recovery phase of Pedersen verifiable secret sharing.

Ideal Bulletin Board. We also need an authenticated ordered bulletin board.

Functionality 4 (Bulletin Board). The bulletin board $\mathcal{F}_{BB}$ running with servers $M_1, \ldots, M_k$, and ideal adversary $\mathcal{S}$ proceeds as follows.

Initialize $D = \emptyset$ and $c = 1$, and repeatedly wait for inputs:

- On input $(\texttt{Write}, s)$ from M_j set $D \leftarrow D \cup \{(c, j, s)\}$, $c \leftarrow c + 1$, and hand $(\texttt{Write}, c, j, s)$ to A.
- On input $(\texttt{Read}, c)$ from M_j such that $(c, l, s) \in D$ hand $(\texttt{Read}, c, l, s)$ to M_j and A.

Realizing Secure Function Evaluation. We now give a secure realization of the secure function evaluation functionality in a hybrid model with the above ideal functionalities.

Protocol 1 (Secure Function Evaluation). The secure function evaluation protocol for f in the $(\mathcal{F}_f^{\mathcal{CS}^B}, \mathcal{F}_{aug}^{\mathcal{CS},\mathcal{CS}^B}, \mathcal{F}_{BB})$-hybrid model running with submittors $S_1, \ldots, S_N$, servers $M_1, \ldots, M_k$, and adversary $\mathcal{S}$ proceeds as follows.

Server M_j:

1. Wait for $(\texttt{PublicKey}, pk^B)$ from $\mathcal{F}_f^{\mathcal{CS}^B}$.
2. Hand $(\texttt{Augment}, pk^B)$ to $\mathcal{F}_{aug}^{\mathcal{CS},\mathcal{CS}^B}$ and wait for $(\texttt{PublicKey}, pk)$ from $\mathcal{F}_{aug}^{\mathcal{CS},\mathcal{CS}^B}$.
3. Write $(\texttt{PublicKey}, pk)$ on $\mathcal{F}_{BB}$.
4. Repeatedly wait for messages and inputs:
   - On an input $(\texttt{Input}, c_i)$ from S_i write (i, c_i) to $\mathcal{F}_{BB}$ and ignore further messages from S_i.
   - On input $(\texttt{RequestOutput})$, write $(\texttt{RequestOutput})$ to $\mathcal{F}_{BB}$. If at least $(k + 1)/2$ servers have written $(\texttt{RequestOutput})$ to $\mathcal{F}_{BB}$, then go to Step 5.
5. Denote by T the contents written to $\mathcal{F}_{BB}$ before the $\lceil (k + 1)/2 \rceil$th occurrence of $(\texttt{RequestOutput})$. Interpret T as a set of the form $\{(j, s)\}$, where j is the identity of the publisher of the information s. Define T_C to consist of the set of (i, c_i) such that $|\{j : (j, (i, c_i)) \in T\}| \geq (k + 1)/2$, i.e., at least $(k + 1)/2$ servers received (i, c_i) from S_i.
6. Hand $(\texttt{Recover})$ to $\mathcal{F}_{aug}^{\mathcal{CS},\mathcal{CS}^B}$ and wait until it returns $(\texttt{PrivateKeyAugmentation}, sk^A)$.
7. Define $C = (\mathsf{Strip}_{pk,sk^A}(c_1), \ldots, \mathsf{Strip}_{pk,sk^A}(c_N))$, where we define $c_i = \mathsf{Enc}_{pk}(0)$ if $(i, c_i) \notin T_C$. Then hand $(\texttt{Input}, C)$ to $\mathcal{F}_f^{\mathcal{CS}^B}$ and wait until it returns $(\texttt{Output}, y)$. Then output $(\texttt{Output}, y)$.

Submittor S_i:

1. Wait for an input $(\texttt{Input}, x_i)$.
2. Wait until $(\texttt{PublicKey}, pk_j)$ is written to $\mathcal{F}_{BB}$ by M_j by $(k + 1)/2$ distinct M_j with identical $pk_j = pk$. Compute $c_i = \mathsf{Enc}_{pk}(x_i)$ and write $(\texttt{Input}, c_i)$ to all M_j.

[3] Again the resulting keys may be biased in a benign way by the adversary.

Proposition 2. *If $\mathcal{CS}$ is a submission secure augmentation of $\mathcal{CS}^B$, then Protocol 1 securely realizes $\mathcal{F}_f$ in the $(\mathcal{F}_f^{\mathcal{CS}^B}, \mathcal{F}_{aug}^{\mathcal{CS},\mathcal{CS}^B}, \mathcal{F}_{BB})$-hybrid model with respect to static adversaries corrupting any minority of the servers and any number of submittors.*

We could extend our proof of concept such that it allows repeated secure function evaluation, but reusing the same instance of $\mathcal{F}_f^{\mathcal{CS}^B}$. To do this we would also have to modify $\mathcal{F}_f$ and $\mathcal{F}_f^{\mathcal{CS}^B}$ such that they keep track of several sessions. We could also extend the protocol such that different sets of submittors uses different augmentations of the basic public key pk^B. This allows moving some of the work involved in stripping ciphertexts to the submission phase of the protocol. We hope the reader agrees that providing details for these extensions would give a much more complicated protocol, but little additional insight.

5 Future Work

In the mix-net application, all messages are free-form. This may not be the case in other applications. It is for example not the case in multi-candidate homomorphic election schemes, e.g., [8], where the submitted messages must be of a special form to encode a valid candidate. It is an interesting question if it is possible to come up with an efficient hash proof system that constrains the set of messages in this way. This would give a very efficient non-interactive submission phase for such election schemes in the standard model.

6 Acknowledgments

I thank Eike Kiltz for helpful discussions, and I thank Ronald Cramer for answering my questions about the relation between their generic scheme and their concrete instantiations.

References

1. M. Abe, R. Cramer, and S. Fehr. Non-interactive distributed-verifier proofs and proving relations among commitments. In *Advances in Cryptology – Asiacrypt 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 206–223. Springer Verlag, 2002.
2. M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *20th ACM Symposium on the Theory of Computing (STOC)*, pages 103–118. ACM Press, 1988.
3. D. Boneh, X. Boyen, and S. Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 226–243. Springer Verlag, 2006.
4. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 136–145. IEEE Computer Society Press, 2001. (Full version at Cryptology ePrint Archive, Report 2000/067, http://eprint.iacr.org, October, 2001.).
5. R. Canetti, O. Goldreich, and S. Halevi. The random oracle model revisited. In *30th ACM Symposium on the Theory of Computing (STOC)*, pages 209–218. ACM Press, 1998.

6. R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In *Advances in Cryptology – Eurocrypt '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 90–106. Springer Verlag, 1999.
7. R. Cramer, I. Damgård, and Y. Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In *2nd Theory of Cryptography Conference (TCC)*, volume 3378 of *Lecture Notes in Computer Science*, pages 342–362. Springer Verlag, 2005.
8. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology – Eurocrypt '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118. Springer Verlag, 1997.
9. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology – Crypto '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer Verlag, 1998.
10. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. http://homepages.cwi.nl/~cramer/, June 1999.
11. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *23rd ACM Symposium on the Theory of Computing (STOC)*, pages 542–552. ACM Press, 1991.
12. P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 427–438. IEEE Computer Society Press, 1987.
13. T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
14. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
15. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
16. J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for np. In *Advances in Cryptology – Eurocrypt 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358. Springer Verlag, 2006.
17. A. Lysyanskaya and C. Peikert. Adaptive security in the threshold setting: From cryptosystems to signature schemes. In *Advances in Cryptology – Asiacrypt 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 331–350. Springer Verlag, 2001.
18. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM Symposium on the Theory of Computing (STOC)*, pages 427–437. ACM Press, 1990.
19. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – Eurocrypt '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer Verlag, 1999.
20. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – Crypto '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer Verlag, 1992.
21. B. Pfitzmann and A. Pfitzmann. How to break the direct RSA-implementation of mixes. In *Advances in Cryptology – Eurocrypt '89*, volume 434 of *Lecture Notes in Computer Science*, pages 373–381. Springer Verlag, 1990.
22. C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology – Crypto '91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer Verlag, 1991.
23. A. Sahai. Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. In *40th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 543–553. IEEE Computer Society Press, 1999.
24. K. Sako and J. Killian. Reciept-free mix-type voting scheme. In *Advances in Cryptology – Eurocrypt '95*, volume 921 of *Lecture Notes in Computer Science*, pages 393–403. Springer Verlag, 1995.
25. V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *Advances in Cryptology – Eurocrypt '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 1–16. Springer Verlag, 1998.

26. D. Wikström. A universally composable mix-net. In *1st Theory of Cryptography Conference (TCC)*, volume 2951 of *Lecture Notes in Computer Science*, pages 315–335. Springer Verlag, 2004.
27. D. Wikström. A sender verifiable mix-net and a new proof of a shuffle. In *Advances in Cryptology – Asiacrypt 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 273–292. Springer Verlag, 2005. (Full version [28]).
28. D. Wikström. A sender verifiable mix-net and a new proof of a shuffle. Cryptology ePrint Archive, Report 2004/137, 2005. http://eprint.iacr.org/.
29. D. Wikström and J. Groth. An adaptively secure mix-net without erasures. In *33rd International Colloquium on Automata, Languages and Programming (ICALP)*, volume 4052 of *Lecture Notes in Computer Science*, pages 276–287. Springer Verlag, 2006.

A Proofs

A.1 Proof of Lemma 1

Denote by $((x_i, \pi_{x,i}), \pi_i)$ the ith query of A and let E_i be the event that $H_k(x_i, \pi_{x,i}) = \pi_i$, $x_i \in X \setminus L$, and $i \leq i_l$ or $x_i \neq x$. Condition on arbitrary fixed values of (x, π_x), $\pi = H_k(x, \pi_x)$, and $\alpha(k)$. Then the conditional probability of the event E_i is negligible by universiality$_2$ of $\mathbb{H}$. Since the fixed values are arbitrary, this holds also without conditioning. Finally, A asks at most a polynomial number of queries and the lemma follows from the union bound. ■

A.2 Proof of Proposition 1

Conceptually, we follow the proof of Cramer and Shoup, but our proof is somewhat simplified, since we ignore the problem of approximating the hash families by efficiently computable hash families.

Denote by $T_b^{(0)}$ the machine that simulates the experiment $\mathsf{Exp}^{\mathsf{sub}-b}_{\mathcal{CS},\mathcal{CS}^B,A}(n)$ with some adversary $A \in \mathrm{PT}^*$, except that when computing the challenge ciphertext $(x, e, \widehat{\pi})$, the two hash proofs π and $\widehat{\pi}$ are computed as $\pi = \mathsf{PEval}_0(\Lambda, k, x) = H_k(x)$ and $\widehat{\pi} = \mathsf{PEval}_1(\Lambda, \widehat{k}_i, x, e) = \widehat{H}_{\widehat{k}_i}(x, e)$, where i is the challenge index chosen by the adversary. By the projectivity of hash proofs this does not change the distribution of the experiment.

We now change $T_b^{(0)}$ step by step until it is independent of b.

Claim 1. Denote by $T_b^{(1)}$ the machine $T_b^{(0)}$ except that x is chosen randomly in $X \setminus L$. Then $|\Pr[T_b^{(0)} = 1] - \Pr[T_b^{(1)} = 1]|$ is negligible.

Proof. Denote by A_{mem} an algorithm that tries to solve the subset membership problem $\mathbb{M}$. It accepts as input (Λ, x), where x either belongs to L or $X \setminus L$. It simulates $T_b^{(0)}$ except that it uses the instance Λ and defines the challenge ciphertext $(x, e, \widehat{\pi})$ using x from its input (Λ, x). Note that A_{mem} is identically distributed to $T_b^{(0)}$ or $T_b^{(1)}$ depending on if $x \in L$ or $x \in X \setminus L$. From the hardness of $\mathbb{M}$ follows that $|\Pr[T_b^{(0)} = 1] - \Pr[T_b^{(1)} = 1]|$ is negligible. □

Denote by $(i_j, (\pi_j, e_j, \widehat{\pi}_j))$ the jth query of A to the decryption oracle $\mathsf{Dec}_{sk_{(\cdot)}}(\cdot)$, and let j_l be the index of the last query before the adversary outputs its choice of challenge index and messages. Denote by $(x, e, \widehat{\pi})$ the challenge ciphertext, and let E be the event that A asks a decryption query $(i_j, (\pi_j, e_j, \widehat{\pi}_j))$ with $\widehat{H}_{\widehat{k}_{i_j}}(x_j, e_j) = \widehat{\pi}_j$, $x_j \in X \setminus L$, and $j \leq j_l$ or $x_j \neq x$, for some index j before it requests $sk^A_{i_j}$ from its $sk^A_{(\cdot)}$-oracle.

Claim 2. $\Pr[E]$ is negligible.

Proof. Let Q be the total number of queries to the augmentation oracle $pk_{(\cdot)}$ made by the adversary. Without loss we assume that the adversary asks the queries $l = 1, \ldots, Q$.

Define A^l_{uni} for $l = 1, \ldots, Q$ to be the machine that simulates $T^{(1)}_b$ and takes part in Experiment 2. The simulation is modified in that $\widehat{s}_l$ is defined as the hash proof key received in Experiment 2, whenever $T^{(1)}_b$ needs to check a hash proof as $\mathsf{PEval}_1(\Lambda, \widehat{k}_l, x_j, e_j) = \widehat{H}_{\widehat{k}_l}(x_j, e_j)$ it simply queries its $\tau_{\widehat{k}_l}(\cdot, \cdot)$-oracle in Experiment 2 with (x_j, e_j) instead, and when it needs to compute $\mathsf{PEval}_1(\Lambda, \widehat{k}_l, x, e) = \widehat{H}_{\widehat{k}_l}(x, e) = \widehat{\pi}$ it outputs (x, e) and waits for $\widehat{H}_{\widehat{k}_l}(x, e) = \widehat{\pi}$ from the experiment instead. The computational universal$_2$ property and the union bound then implies the claim.

Note that the computational universal$_2$ property can be applied despite that the experiment reveals private hash proof keys, since by definition of submission security the adversary only wins if it never asks a decryption query after this point. This observation is the only essential change to the original proof. □

Denote by $T^{(2)}_b$ the machine $T^{(1)}_b$, except that it outputs $\perp$ if the event E occurs. The machine $T^{(2)}_b$ may not be efficient, but this does not matter since the remainder of the argument is statistical.

Claim 3. Denote by $T^{(3)}_b$ the machine $T^{(2)}_b$ except that in the computation of the challenge ciphertext $(x, e, \widehat{\pi})$, π is chosen randomly in Π. Then $|\Pr[T^{(2)}_b = 1] - \Pr[T^{(3)}_b = 1]|$ is negligible.

Proof. Consider an arbitrary fixed instance Λ of the subset membership problem and an arbitrary fixed random string of the experiment conditioned on the event $\bar{E}$. Define a function $f : X \times S \times \Pi \to \{0, 1\}$ as follows. Let $f(x, \alpha(k), \pi)$ simulate $T^{(2)}_b$ except that the input parameters are used in the computation of the challenge ciphertext. Note that f exists, since $T^{(2)}_b$ outputs $\perp$ if the event E occurs and $\alpha(k)$ determines H_k on L by the projective property of $\mathbb{H}$, so the answers of all queries are determined by $\alpha(k)$. When $k \in K$, $x \in X$, and $\pi \in \Pi$ are randomly chosen, $f(x, \alpha(k), H_k(x))$ is identically distributed to $T^{(2)}_b$ and $f(x, \alpha(k), \pi)$ is identically distributed to $T^{(3)}_b$. The claim now follows from the smoothness of $\mathbb{P}$.

Conclusion of Proof of the Proposition. To conclude the proof of the proposition we simply note that the distributions of $T^{(3)}_0$ and $T^{(3)}_1$ are identical. The claims above now imply that $|\Pr[T^{(0)}_0 = 1] - \Pr[T^{(0)}_1 = 1]|$ is negligible. ■

A.3 Proof of Proposition 2

Suppose we are given an adversary $\mathcal{A}$ in the hybrid model such that for every ideal adversary $\mathcal{S}$, there exists an environment $\mathcal{Z}$ such that the difference between the output of $\mathcal{Z}$ in the two models is not negligible. We show that such an adversary contradicts the submission security of the cryptosystem $\mathcal{CS}$ derived from the cryptosystem $\mathcal{CS}^B$. We first describe an ideal adversary and then we reach a contradiction.

The ideal adversary. We first describe an ideal adversary $\mathcal{S}$ that simulates the hybrid protocol using $\mathcal{A}$ as a black-box, but with a number of modifications. Denote by J and I the indices of servers M_j and submittors S_i respectively corrupted by the adversary $\mathcal{A}$ in $\mathcal{S}$'s simulation. $\mathcal{S}$ corrupts the corresponding dummy parties in the ideal model. Then it proceeds with its simulation except for the following changes:

Short-circuit communication. We ensure that the environment essentially communicate directly to the simulated corrupted parties and the adversary as follows:

- Any input to a corrupted dummy submittor $\tilde{S}_i$ with $i \in I$ is forwarded to the corresponding corrupted simulated submittor S_i, and any output from the corrupted simulated submittor S_i is forwarded to the corresponding dummy submittor $\tilde{S}_i$.
- Any input to a dummy server $\tilde{M}_j$ with $j \in J$ is forwarded to the corresponding simulated submittor M_j, and any output from the corrupted simulated server M_j is forwarded to the corresponding dummy server $\tilde{M}_j$.
- Any input to $\mathcal{S}$ from the environment $\mathcal{Z}$ is forwarded to $\mathcal{A}$ and any output from $\mathcal{A}$ is forwarded to the environment $\mathcal{Z}$.

Extraction from corrupted submittors. The ideal adversary must extract the inputs used by corrupted submittors in the simulation and hand these inputs to the ideal functionality $\mathcal{F}_f$. This is done as follows. When a pair (i, c_i) is written to $\mathcal{F}_{BB}$ which implies that (i, c_i) will belong to the set T_C the simulation of $\mathcal{F}_{BB}$ is interrupted. Then $\mathcal{S}$ computes $x_i = \mathsf{Dec}_{sk}(c_i)$ and instructs the dummy submittor $\tilde{S}_i$ to hand x_i to $\mathcal{F}_f$. When $\mathcal{F}_f$ hands $(\texttt{Input}, x_i)$ to $\mathcal{S}$ the simulation of $\mathcal{F}_{BB}$ is resumed.

Simulation of honest submittors. When a dummy submittor hands an input to the ideal functionality $\mathcal{F}_f$, the ideal adversary must make sure that it appears as if the corresponding simulated submittor submitted the same input in the simulated hybrid protocol. This is done as follows. When $\mathcal{S}$ is handed $(\texttt{Input}, i)$ with $i \notin I$ from $\mathcal{F}_f$ it sets $x'_i = 0$ and inputs $(\texttt{Input}, x'_i)$ to the simulated submittor S_i. Note that x'_i most likely is different from the true value x_i submitted by S_i to $\mathcal{F}_f$.

Extraction from corrupted servers. When a corrupted server requests the computation of the function in the simulated hybrid protocol the simulator must make sure that the corresponding dummy server requests that the ideal functionality $\mathcal{F}_f$ computes the function f as well. This is done as follows. When M_j with $j \in J$ writes $(\texttt{RequestOutput})$ to $\mathcal{F}_{BB}$ the simulation of $\mathcal{F}_{BB}$ is interrupted. Then $\mathcal{S}$ instructs $\tilde{M}_j$ to hand $(\texttt{RequestOutput})$ to $\mathcal{F}_f$. When $\mathcal{F}_f$ hands $(\texttt{RequestOutput}, j)$ to $\mathcal{S}$, the simulation of $\mathcal{F}_{BB}$ is continued.

Simulation of honest servers. When an honest server requests the computation of f from the ideal functionality $\mathcal{F}_f$, the corresponding simulated server must request the computation of f in the real protocol. This is done as follows. When $\mathcal{S}$ is handed ($\texttt{RequestOutput}, j$) with $j \notin J$, it inputs ($\texttt{RequestOutput}$) to M_j in its simulation.

Reaching a contradiction. Denote by $\mathcal{Z}$ the environment that is able distinguish the hybrid model from the real model that does not have negligible advantage. Denote by W_0 its output when executing the ideal model with $\mathcal{S}$, and denote by W_1 its output when executing in the hybrid model with $\mathcal{A}$.

Denote by H_0 the algorithm that simulates an execution in the ideal model with $\mathcal{S}$ and $\mathcal{Z}$. Define H_l to be H_0 except that when simulating the submission from an honest submittor S_i with $i \leq l$, it replaces the input $x'_i = 0$, by the true value x_i that was submitted by $\tilde{S}_i$ to $\mathcal{F}_f$. The true value can be extracted from $\mathcal{F}_f$, since H_l simulates $\mathcal{F}_f$ and can look at its internal data.

The result of this is that the output of H_N is *identically* distributed to W_1. This can be seen by inspection. More precisely:

1. An input extracted from a corrupt submittor is given to $\mathcal{F}_f$ only if the corresponding output is guaranteed to be used in the computation of the output in the protocol.
2. An input is given to a simulated honest submittor only if the corresponding honest dummy submittor hands an input to $\mathcal{F}_f$ in the ideal model. Since the correct inputs are used in H_N the submitted cleartexts are also identical in the two models.
3. The output is requested from $\mathcal{F}_f$ by a corrupted dummy server in the ideal model only if it is requested by the corresponding server in the protocol.
4. The output is requested in the protocol by an honest simulated server only if the corresponding dummy server requests the output from $\mathcal{F}_f$.
5. An output is generated in both models exactly when some $\lceil (k+1)/2 \rceil$th server have requested the output, and the output is defined in both models as all messages/plaintexts submitted until the $\lceil (k+1)/2 \rceil$th server requests the output.

A hybrid argument now implies that $|\Pr[H_{i-1} = 1] - \Pr[H_i = 1]|$ is not negligible for some fixed i.

Finally, we define B to be an adversary in the submission security experiment, Experiment 1. It simulates H_{i-1} with the following changes. It instructs $\mathcal{F}_f^{\mathcal{CS}^B}$ to use the public key pk^B it receives in Experiment 1, instead of generating this public key by itself. It hands the query 1 to its $pk_{(\cdot)}$-oracle and instructs $\mathcal{F}_{aug}^{\mathcal{CS},\mathcal{CS}^B}$ to use the reply $pk_1 = (pk_1^A : pk^B)$ instead of generating (pk_1^A, sk_1^A) by itself. When $\mathcal{F}_f^{\mathcal{CS}^B}$ needs to decrypt a ciphertext c_j, B hands the query $(1, c_j)$ to its $\mathsf{Dec}_{sk_{(\cdot)}}(\cdot)$-oracle and instructs $\mathcal{F}_f^{\mathcal{CS}^B}$ to use the result x_j as if it decrypted it itself. When $\mathcal{F}_{aug}^{\mathcal{CS},\mathcal{CS}^B}$ must output the secret augmentation key, B hands the query 1 to its $sk_{(\cdot)}^A$-oracle and instructs $\mathcal{F}_{aug}^{\mathcal{CS},\mathcal{CS}^B}$ to use the reply sk_1^A as if it had generated it by itself.

The final change to H_{i-1} is that when it is about to compute the simulated ciphertext c_i it instead hands $(0, x_i)$ to Experiment 1, and uses c_i as before in the continued simulation.

The output of B is identically distributed to the output of H_{i-1} or H_i respectively depending on if c_i is an encryption of 0 or x_i. Thus, B breaks the submission security of $\mathcal{CS}$, which is a contradiction. ■