Encrypting Messages for Incomplete Chains of Certificates

Sanjit Chatterjee*, Deepak Garg†, Aniket Kate‡ and Tobias Theobald§

*IISc, India sanjit@csa.iisc.ernet.in
†MPI-SWS, Germany dg@mpi-sws.org
‡Purdue University, USA aniket@purdue.edu
§Healthcast S.A., Germany tobias@health-cast.com

***Abstract*—A public key infrastructure (PKI) binds public keys to the identities of their respective owners. It employs certificate authorities or a web of trust over social links to transitively build cryptographic trust across parties in the form of chains of certificates. In existing PKIs, Alice cannot send a message to Bob confidentially until a complete chain of trust from Alice to Bob exists. We observe that this temporal restriction—which may be severely limiting in some contexts like whistleblowing—can be eliminated by combining webs of trust with concepts from hierarchical identity-based encryption.**

Specifically, we present a novel protocol that allows Alice to securely send a message to Bob, binding to any chain of social links, with the property that Bob can decrypt the message only after trust has been established on all links in the chain. This trust may be established either before or after Alice has sent the message, and it may be established in any order on the links. We prove the protocol's security relative to an ideal functionality, develop a prototypical implementation and evaluate the implementation's performance for a realistic environment obtained by harvesting data from an existing web of trust. We observe that our protocol is fast enough to be used in practice.

I. Introduction

Establishing cryptographic trust in electronic communication is a non-trivial task. When the communicating parties do not exchange key material offline or when a client does not verify key material received from a server in an offline meeting, it is very hard to ascertain that the connection is authentic, i.e., that no adversary can eavesdrop or manipulate messages. One approach to this problem is using a Public-Key Infrastructure (PKI), in which a trusted third party, called a Certificate Authority (CA) verifies a person's identity offline to ensure that the person is actually who they claim to be and then creates a certificate for that person's public key. This signature can afterwards be verified by anyone using the CA's public key and trusting the CA. However, the problem with this approach is the requirement of a trusted third party, which, as a so-called trust anchor, is a very natural entry point for an attacker who wants to compromise the system through vulnerabilities [10], [11] or legal pressure.

Moreover, employing a CA's service may not always be financially feasible for end-users. As a result, several large-scale online services such as Apple iMessage and WhatsApp now offer end-to-end encrypted communication capabilities to their users employing a centralized directory of public keys that the online service maintains. As these public key services remain vulnerable to attacks [1] similar to those on CAs, some providers are considering use of systems like CONIKS [23] that protect confidentiality as long as at least one of several centralized service providers remains uncompromised.

The email and file encryption tool Pretty Good Privacy (PGP), instead, relies on a completely distributed trust system that is not based on centralized CAs or service providers, but on the fact that people tend to know and reliably authenticate their friends. This fact is used to obtain the necessary trust in public keys of people transitively along the is-friend relation, resulting in a so-called Web of Trust (WoT). In a WoT, everyone implicitly acts as a CA for their immediate friends, and signs their keys. A user trusts everyone to properly verify their friends. Because signatures on keys are published, every user can try to find a trustworthy path through this web of trust to the message's recipient and, if such a path exists, the sender can be reasonably sure that the communication is secure. Although WoT has known usability issues, there is renewed interest in WoT in the community, e.g., in the form of the recent rebooting-the-web-of-trust effort [2].

In all public key (or certificate) verification systems described above as well as in most other peer-to-peer communication systems [9], [16], [25], [31], a user can send a signed message to any receiver even before a trusted from the sender to the receiver exists. (Of course, the receiver can authenticate the message only after the trusted path exists.) This is, however, not possible for confidential/encrypted communication: a trusted path from the sender to the recipient is necessary *before* the sender can securely encrypt a message for the recipient. There is no way for the sender to send the message and for the recipient to start participating in the system only afterwards (perhaps, specifically to decrypt the message). While this might seem like a problem that can be solved by telling the recipient to participate in the system and waiting for them to do so before sending the message, often the circumstances of the sender make it too risky to wait. The problem is particularly relevant for contexts like whistleblowing, where the adversary is very strong and the whistleblower may not have a second opportunity to contact the intended recipient or may be in constant danger of being persecuted. As result, there is real-world need for a solution for PKI/WoT where a sender can encrypt a message before a trusted path to the recipient is formed.

Contributions. Ours is the first work to address the problem of allowing confidential communication in a PKI/WoT *before* bootstrapping trust (or link verification) between the communi-

cating parties. In particular, we propose an enhanced PKI certification process that incorporates, in addition to public keys, a form of hierarchical identity-based encryption (HIBE) [7], [17], [19], [20]. In HIBE, a sender can encrypt a message to an arbitrary, subsequently verifiable identifier of the recipient (e.g., the recipient's email address or social security number). The recipient can decrypt the message by establishing a chain of trust leading to a private key generator (PKG), possibly after the message was encrypted. Our scheme employs the HIBE idea in a WoT/PKI setting. It functions like an ordinary WoT, but allows a sender to send an encrypted message to a recipient to whom a verified path may *not* already exist, and even in cases when the recipient is not already a part of the system. The scheme guarantees that the recipient can decrypt the message only after a verified path from the sender to the recipient has been established. With our scheme in place, a sender like a whistleblower can send her message immediately and exit the system without waiting for the recipient to join the system. Owing to its dependence on both identity-based encryption and webs of trust, we call our scheme an Identity Web of Trust (IWoT).

To send a message in an IWoT, the sender chooses a path to the recipient. The path is split into subpaths of contiguous links that have already been verified and contiguous links that have not already been verified. Roughly speaking, encryption is layered, using one HIBE encryption for every unverified subpath. Later, when individuals along the non-verified subpaths verify each other, they execute a designated authentication protocol that reveals evidence of the verification publicly. After all links on the path have been verified, the recipient can combine the public evidence to generate enough keys to decrypt the message. The key challenges in designing the scheme lie in coming up with public evidence of verification of a link that, when combined along a path, allows for decryption, and designing the protocol in a way that allows the links along a path to be verified in *any* order.

We define the security of our scheme as an ideal functionality, and formally prove the security of the protocol in the simulation-based security paradigm [21]. We have also implemented a Java library that provides all necessary IWoT functions and thoroughly evaluated its performance for a realistic environment built by harvesting an existing WoT graph from the SKS OpenPGP key server [24]. We also describe extensions that employ multiple encryption paths for better confidentiality and availability, offer source authentication, and provide ciphertext anonymity.

Organization. Section II provides some background about webs of trust and identity-based cryptography. In Section III, we present our problem statement and offer an overview of IWoT. We describe the detailed cryptographic construction in Section IV, and provide an ideal functionality and show our scheme secure relative to the functionality assuming standard security for the underlying HIBE and public-key schemes in Section V. We implement our scheme and evaluate the overheads of its individual operations in Section VI. We discuss security, privacy, and availability enhancements in Section VII.

II. Background

This section provides necessary background on webs of trust and identity-based encryption, both of which we use as primitives in our work.

A. Web of Trust

A public key infrastructure (PKI) binds public keys to the identities of their respective owners through a chain of certificates. Although, in the industry and the government, trustworthy binding is typically established through a commercial engagement with a trusted certificate authority (CA), a web of trust offers an interesting alternative by employing transitive trust over social links.

A web of trust (WoT) is an overlay on an underlying graph of trust relationships with the goal of verifying public keys of the nodes (parties) in the underlying graph. When party B gets to know party A (through friendship, professional contact, or through offline document verification), it can sign A's public key with its own private key and publish the signature. If a third-party C knows B's public key and it trusts B to have authenticated A correctly, it can now trust A's public key to actually be A's. This trust in A's public key can be extended transitively. More importantly, even if a single chain from a verifier to a verified party is not entirely trusted by the verifier, the verifier might still have substantial confidence in the public key of the verified party through several distinct chains leading from the verifier to the verified party. PGP [3] is a widely employed implementation of webs of trust. Emerging research on blockchains and the development community are also drawing fresh attention to the WoT concept [2].

In our work, we use WoT as the basis for our solution. However, our solution remains equally applicable to CA-based as well as service provider-based PKIs.

B. Identity-based Encryption

In public key cryptography (PKC), every individual generates a private key and a public key. Not only must this generation of keys be done before any messages are encrypted, but the encryptor must also know the authenticated public key of the recipient before performing the encryption. This can be a strong requirement, which is not always satisfied in practice. Identity-based encryption (IBE) [8], [12], [28] alleviates this problem. In IBE, an encryptor uses any verifiable *identity* of the recipient (e.g., the recipient's email address or the recipient's social security number) to generate the recipient's public key for encryption. The only information it requires is the identity of the recipient (which may be unauthenticated) and some public parameters belonging to a trusted third-party, called a private key generator or PKG. These public parameters are published upfront. To decrypt, the recipient contacts the PKG and authenticates with its identity. The PKG then uses private parameters corresponding to the published public parameters and the identity to generate the recipient's decryption key. Figure 1 shows how IBE works. The primary advantage of IBE, as opposed to PKC, is that the authentication of the recipient (to the PKG) can occur either before or after the encryption is made.

C. Hierarchical identity-based encryption—HIBE

HIBE [7], [17], [19], [20] is a generalization of IBE that supports hierarchical namespaces of identities and allows a party with identity I to act as the PKG for the entire namespace hierarchy under I. Specifically, a HIBE has a root identity R, which acts as the PKG for the whole system. Immediately below R are the identities $(R, A), (R, B), \ldots$. The party with identity (R, A) can act as the PKG for all identities that extend (R, A), i.e., for the identities (R, A, A'), (R, A, B'), etc.

Technically, any party with the decryption key for the identity I can *generate* the decryption keys for all identities that extend I. We denote the decryption key of identity I that decrypts messages encrypted using I's HIBE master encryption key pk_H with $d_{pk_H}(I)$. The system is bootstrapped by R generating its own decryption key $d_{pk_H}(R)$. It then generates $d_{pk_H}(R, A)$ and gives it to the party with identity (R, A), gives $d_{pk_H}(R, B)$ to the party with identity (R, B), etc. Either R or (R, A) can then generate the key $d_{pk_H}(R, A, A')$ and provide it to the party with identity (R, A, A') and so on. Figure 2 illustrates how HIBE works.

Importantly, in a HIBE, a party with identity I is a *key escrow* for all identities that extend I, in the sense that it can generate the keys of all parties that extend I. Hence, I must be trusted by all parties that extend I.

III. Problem Statement and Solution

This section presents our problem statement, threat model, and our solution—identity webs of trust (IWoTs).

A. Problem Statement

We work in the web of trust (WoT) setting, where parties (principals) develop mutual trust relationships and use them

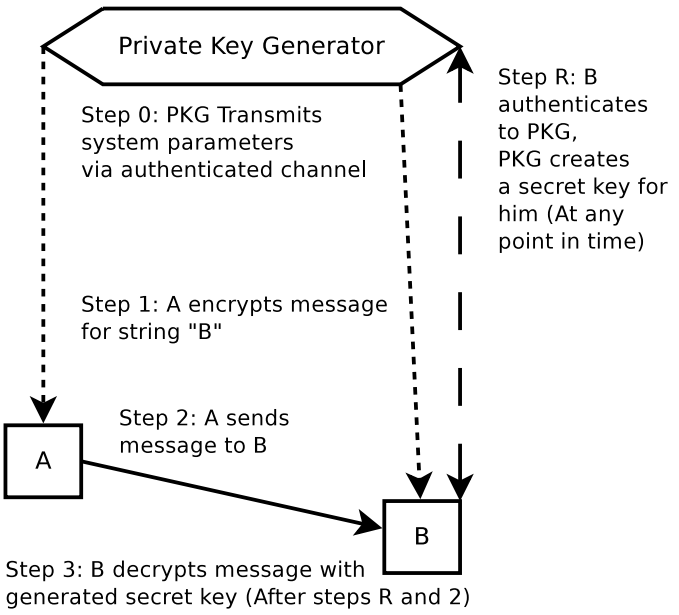

Fig. 1. A simple example of how IBE works. After everyone has the PKG's public parameters, A can encrypt a message for B's identity string "B". B can (at any point in time, before or after) retrieve its private key from the PKG and afterwards decrypt the message.

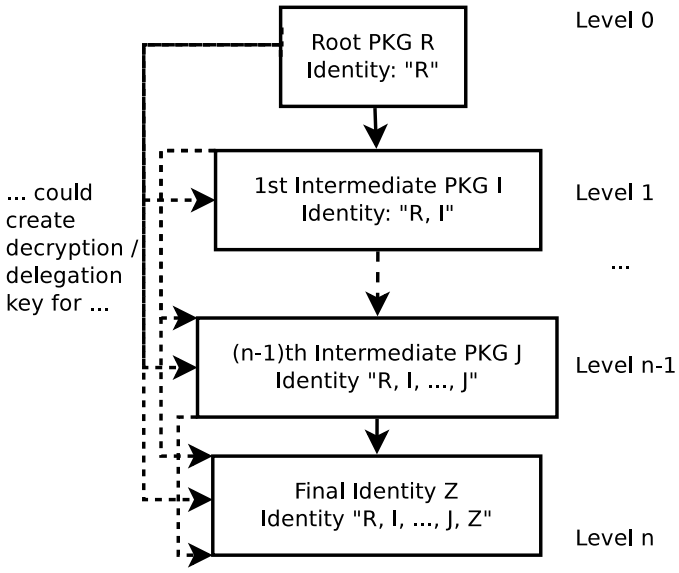

Fig. 2. A simple example of how HIBE works. R is the root private key generator (PKG) identity.

to authenticate their identities with their cryptographic keys without any central trusted third party. Each party is assumed to have a named identity. As explained earlier, in the conventional WoT settings, a party with identity A can create a confidential message for another party with identity B *only after* all links on at least one chain from A to B have been authenticated. We wish to allow A to create a confidential message for B *even before* an authenticated chain between A and B has fully emerged.

We assume that A is able to pick a chain of identities from itself to the recipient B, which A expects will become fully authenticated in due course. We wish to allow A to encrypt with a *binding* to this chain such that B can decrypt the message only after all links of the chosen chain have been authenticated. Assuming that all links authenticate correctly, this implies that only the intended B receives the cleartext message. Moreover, we want B to be able to decrypt the message independent of the order in which links between A and B are authenticated.

Threat model. As in a WoT system, an attacker can create fake (or Sybil) parties in the network, and can try to impersonate honest parties by proclaiming their identities. It can also force a party it controls to authenticate any other party as any identity. Given the incomplete nature of our WoT graphs, encryptors may unknowingly choose one or more impersonator parties instead of real identity owners. We guarantee message confidentiality in this strong adversarial setting. We assume all parties have access to all material published by all other honest parties over a publicly accessible, append-only bulletin board.

It is fundamentally impossible to send a confidential message to a receiver without knowing some symmetric/asymmetric key associated with the receiver. Therefore, we cannot always address the *last mile limitation*, where, in a directed chain of links to the receiver, the last unauthenticated party can compromise the confidentiality of the message. Nevertheless, when either the recipient's public key (even

unauthenticated) or multiple chains are available at the time of encryption, we can mitigate this issue.

Non-goals. We are concerned only with the security of messages; the question of physical transmission and delivery of messages, as well as the question of denial-of-service attacks, are orthogonal and not the subject of this paper. Moreover, we do not address the problem of how a sender picks a path that will be authenticated in the future. In practice, the sender will pick a path based on her knowledge of the geographic location of parties on the path, upcoming rendezvous, and social contacts. If a strong adversary compromises some honest parties and obtains their private keys, we cannot always guarantee confidentiality of messages sent over trust paths passing through the compromised parties.

Finally, similar to a WoT, it is not our goal to protect the privacy of social WoT graphs and other related meta-data, and we expect the concerned users to employ pseudonyms instead of their real identities to get some privacy protection. Nevertheless, in Section VII, we extend our scheme to prevent an adversary from learning the sender, the recipient and the chosen verification path from the ciphertext itself.

B. The Solution: Identity Web of Trust (IWoT)

IWoTs solve the problem described above by combining HIBE with a WoT. To bring out the key design decisions, we consider a sequence of scenarios of increasing complexity and show how encryption, decryption and link authentication work in each scenario. We introduce some notation first.

Notation. Uppercase letters A, B, C, etc. stand for identities of parties. $pk_S(N)$ and $sk_S(N)$ denote the public (verification) and private (signing) keys of N. Similarly, $pk_E(N)$ and $sk_E(N)$ stand for the public (encryption) and private (decryption) keys of a party with identity N. In some cases, our protocol requires an identity to act as a HIBE root; $pk_H(N)$ and $sk_H(N)$ denote, respectively, the HIBE master public and secret (decryption) keys of identity N. All public keys are initially unauthenticated. They are authenticated as the web of trust develops. $d_{pk_H(A)}(A, \ldots, Z)$ denotes the HIBE decryption key generated by the PKG A (i.e., with master public key $pk_H(A)$) for the hierarchical identity $(A, \ldots, Z)$. $Enc(pk_E(N), m)$ denotes the public key encryption of m with key $pk_E(N)$, $Enc(k, m)$ denotes the encryption of m with the symmetric key k, and $Enc(pk_H(A), (A, \ldots, Z), m)$ denotes the HIBE encryption of m for the decryption key $d_{pk_H(A)}(A, \ldots, Z)$, with the public HIBE parameters of A. In some places, we subscript the function Enc with E, S or H (public-key, symmetric or HIBE, respectively) to visually emphasize the kind of encryption.

Link authentication. When a message is sent in an IWoT, some links in the network have already been authenticated, whereas other links have not been authenticated (yet). We call the former *strong links* and the latter *weak links*. A strong link from C to D is written $C \Rightarrow D$ and a weak link from C to D is written $C \rightarrow D$. For simplicity, we assume that if $C \Rightarrow D$, then C trusts D completely for the purpose of our protocol. When C authenticates D, i.e., when D proves to C that it owns the identity D, C signs D's three public keys—verification, encryption and HIBE—and posts the signatures on a *public bulletin board*, which is accessible to all parties. C also performs some other operations related to HIBE, which we describe gradually. Conceptually, the authentication changes $C \rightarrow D$ to $C \Rightarrow D$.

The required bulletin board can be implemented starting from a reliable broadcast protocol [30] or over a permissioned or permissionless blockchain [15], [26]. However, for the purpose of our discussion, we treat it as a standard, ideal bulletin board $\mathcal{F}_{BB}$ [32].

A path or chain is a sequence of identities linked by weak or strong links. We write $\Rightarrow^*$ and $\Rightarrow^+$ for the reflexive-transitive and transitive closures of $\Rightarrow$ (similarly for $\rightarrow$). We often conflate a strong path $X_1 \Rightarrow X_2 \Rightarrow \ldots X_n$ with the HIBE identity $X_1, X_2, \ldots, X_n$ (similarly for $\rightarrow$, and any composition of $\rightarrow$ and $\Rightarrow$). Note that every party can determine which strong chains $X \Rightarrow^* Y$ exist by looking at the signatures on the bulletin board.

To encrypt a message, the sender picks a path from itself to the recipient, which it expects will eventually become strong (the path may already be strong). The encryption of the message binds to that path and our protocol guarantees that the recipient will be able to decrypt the message only after all links on the path have been authenticated, i.e., become strong (in any temporal order). For efficiency, the sender should try to choose a path with a low number of weak links. Additionally, the sender should try to minimize the maximum length of contiguous weak link subchains on the path it chooses, as this reduces the number of HIBE keys generated during link authentication.

Scenario 1: Strong chains (public-key encryption). Suppose A wants to encrypt a message for B along a chain of strong links only (i.e., a chain of the form $A \Rightarrow^+ B$). In this case, our protocol degenerates to a conventional protocol for encryption over WoTs: A verifies B's encryption public key, $pk_E(B)$, using the signatures (along the chain) on the bulletin board and uses this public key to encrypt the message. Since there is a complete chain of trust from A to B, B's public key must be authentic and, hence, only B can decrypt the message.

Scenario 2: Weak chains (HIBE). Next, we consider the extreme scenario where A encrypts a message for B along a chain of weak links only (a chain of the form $A \rightarrow^+ B$). In this case, A does not have access to B's authenticated public encryption key. So, our protocol uses HIBE instead of public key encryption: A HIBE-encrypts the message m for the identity that equals the path. So, the ciphertext is $Enc(pk_H(A), (A \rightarrow^+ B), m)$.

Of course, B should be able to decrypt the ciphertext (i.e., obtain the decryption key $d_{pk_H(A)}(A \rightarrow^+ B)$) (only) after all links in $A \rightarrow^+ B$ have been authenticated. For this, we require that when any C authenticates any D, for every HIBE decryption key $d_{pk_H(P_1)}(P_1, \ldots, P_n, C)$ that C possesses and for every $D \Rightarrow^* E$ that exists (on the bulletin board), C also generate the key $d_{pk_H(P_1)}(P_1, \ldots, P_n, C, D \rightarrow^* E)$ and post it *encrypted with* $pk_E(E)$ on the bulletin board. It is easily seen that this authentication protocol provides the following property:

Property 1. The ciphertext $Enc(pk_E(Y), d_{pk_H(X)}(X \rightarrow^+ Y))$ exists on the bulletin board after $X \rightarrow^+ Y$ has become strong.

Importantly, this property holds *independent of the order* in which the links on $X \rightarrow^+ Y$ become strong. Returning to our scenario, once the path $A \rightarrow^+ B$ becomes strong, B gets the key $d_{pk_H(A)}(A \rightarrow^+ B)$, allowing it to decrypt the message.

The inherent last-mile limitation. It is impossible to encrypt a message for a receiver B if the encryptor A does not have access to some authenticated/unauthenticated key of the receiver B. IBE and HIBE overcome this impossibility by introducing a key escrow: all parties on the path $A \rightarrow^+ B$ can generate the decryption key $d_{pk_H(A)}(A \rightarrow^+ B)$ and decrypt the message. They are, therefore, *key escrows*—they must be trusted to not generate this key or not use it.

This key escrow problem can be overcome in one of two ways. First, if even an unauthenticated public key for B is available to A at the time of encryption, then A, in the form of certificateless encryption [4], can protect the ciphertext with an additional layer of public key encryption to that key. This eliminates the key escrow problem completely and can be highly effective in services such as Apple iMessage and WhatsApp where users generate the public key pairs right when they register. Second, if B's public key is not known to A at the time of encryption, A can resort to more involved encryption schemes using multiple paths to mitigate the escrow problem (see Section VII-A). This discussion about mitigation of the escrow problem applies to all subsequent scenarios, although we do not mention it explicitly again.

Scenario 3: Weak-strong chains. We generalize scenario 2 to the case where encryption is bound to a chain of weak links followed optionally by strong links. This chain has the form $A \rightarrow^+ D \Rightarrow^* B$. In this case, IWoT encryption treats $D \Rightarrow^* B$ as *weak*, and A encrypts for the chain $A \rightarrow^* D \Rightarrow^* B$. The ciphertext is $Enc(pk_H(A), (A \rightarrow^+ D \Rightarrow^* B), m)$. Due to Property 1, B can decrypt (only) after all links on the chain become strong.

Scenario 4: Weak-strong-weak-strong chains (layering and finalized keys). Next, consider a scenario where the chain pattern weak-strong of Scenario 3 repeats once, i.e., where A wants to encrypt to a chain of the form $A \rightarrow^+ C \Rightarrow^+ D \rightarrow^+ E \Rightarrow^* B$. An obvious solution in this case is to treat the whole chain as weak and encrypt using a single HIBE to the decryption key $d_{pk_H(A)}(A \rightarrow^+ C \Rightarrow^+ D \rightarrow^+ E \Rightarrow^* B)$. As noted at the end of Scenario 2, this would make all identities on the chain key escrows. We show here that we can do better: We can remove all identities in the chain $A \rightarrow^+ C \Rightarrow^+$ (this chain excludes D) from the escrow set, leaving only the identities in $D \rightarrow^+ E \Rightarrow^* B$ as escrows. To obtain this stronger property, we use two layers of encryption. We start with a (incorrect) strawman solution, which we refine to a correct solution.

In the strawman solution, the plaintext m is first HIBE encrypted to the key $d_{pk_H(A)}(A \rightarrow^+ C \Rightarrow^+ D)$ and the resulting ciphertext is then HIBE encrypted to the key $d_{pk_H(D)}(D \rightarrow^+ E \Rightarrow^* B)$. The intuition is that identities on the path $A \rightarrow^+ C \Rightarrow^+$ (excluding D) do not have access to D's HIBE secret key, so they cannot generate the key $d_{pk_H(D)}(D \rightarrow^+ E \Rightarrow^+ B)$ and, hence, cannot decrypt the message. However, this layered encryption has a problem: The decryption key $d_{pk_H(A)}(A \rightarrow^+ C \Rightarrow^+ D)$ is D's decryption key, not the recipient B's, so B won't ever have access to it! Specifically, if we look at Property 1, we see that this key will be posted on the bulletin board encrypted with D's public encryption key, not B's. In fact, giving B access to the key $d_{pk_H(A)}(A \rightarrow^+ C \Rightarrow^+ D)$ would be insecure because there could be other messages that were intended *for* D, which B would be able to decrypt if it had access to the key.

To resolve this dilemma, let us examine what property we intend to obtain by including a layer of encryption to the key $d_{pk_H(A)}(A \rightarrow^+ C \Rightarrow^+ D)$. This layer does not have to provide confidentiality, which is already provided by the second layer that encrypts to the recipient B's decryption key, $d_{pk_H(D)}(D \rightarrow^+ E \Rightarrow^* B)$. Instead, this layer provides *authentication*: We want to ensure that the key to decrypt the layer will exist only after the chain $A \rightarrow^+ C$ becomes strong. This observation provides the main insight to our solution: We introduce additional HIBE decryption keys whose only purpose is to prove the existence of strong links. Since these keys do not provide confidentiality, they can be released publicly. Specifically, we introduce a variant of each identity F, written $F!$. Our invariant is the following:

Property 2. $d_{pk_H(X)}(X, Y, \ldots, Z, F!)$ is *publicly available* on the bulletin board after the entire path $X \rightarrow Y \ldots Z \rightarrow F$ has become strong.

$d_{pk_H(X)}(X, Y, \ldots, Z, F!)$ is generated in the same way and at the same time as $d_{pk_H(X)}(X, Y, \ldots, Z, F)$. The differences are that (a) $d_{pk_H(X)}(X, Y, \ldots, Z, F!)$ is immediately posted to the bulletin board in the cleartext (so anyone can use it) and (b) it is never "extended" by honest parties, e.g., we never get a key of the form $d_{pk_H(X)}(X, \ldots, F!, G, \ldots)$ if each of $X, ..., F$ is honest. Since identities like $F!$ are never extended, we call them *finalized* identities and call keys like $d_{pk_H(X)}(X, \ldots, Z, F!)$ finalized keys.

We change our layered encryption to use a finalized key: To encrypt to the path $A \rightarrow^+ C \Rightarrow^+ D \rightarrow^+ E \Rightarrow^* B$, A uses two HIBE encryption layers, one to the key $d_{pk_H(A)}(A \rightarrow^+ C \Rightarrow^+ D!)$ (note the ! after D) and the other to the key $d_{pk_H(D)}(D \rightarrow^+ E \Rightarrow^* B)$ (note the absence of ! after B). After both $A \rightarrow^+ C$ and $D \rightarrow^+ E$ have become strong, the key $d_{pk_H(A)}(A \rightarrow^+ C \Rightarrow^+ D!)$ is on the bulletin board in the clear (Property 2) and the key $d_{pk_H(D)}(D \rightarrow^+ E \Rightarrow^* B)$ is on the bulletin board encrypted with B's public encryption key (Property 1), so B can get both keys and decrypt the ciphertext. It should be clear that identities on the path $A \rightarrow^+ C \Rightarrow^+$ (excluding D) cannot decrypt the message ciphertext because they cannot generate or obtain the key $d_{pk_H(D)}(D \rightarrow^+ E \Rightarrow^* B)$. Hence, they are not escrows.

We point out two more subtleties. First, in order to create the first encryption layer to the key $d_{pk_H(D)}(D \rightarrow^+ E \Rightarrow^* B)$, A must have access to D's public HIBE key. Since D lies at the end of a strong chain ($C \Rightarrow^+ D$), its HIBE public key can be obtained from the bulletin board but since C has not been authenticated by A (C lies at the end of the *weak* chain $A \rightarrow^+ C$), there is no guarantee that D's HIBE public key

on the bulletin board, say pk, is authentic. Consequently, A's ciphertext must use pk in a way that if pk turns out to be non-authentic, message decryption is not possible. One simple way to do this, which we adopt here, is to add the key pk to the end of the finalized identity $D!$. The first layer of encryption is to the key $d_{pk_H(A)}(A \rightarrow^+ C \Rightarrow^+ D!pk)$. The decryption key generated by authentication of the chain $A \rightarrow^+ C \Rightarrow^+ D$ is $d_{pk_H(A)}(A \rightarrow^+ C \Rightarrow^+ D!pk_H(D))$, so, if $pk \neq pk_H(D)$, no one can decrypt the message ciphertext. Property 2 can be revised to reflect this change to the protocol.

Property 2'. $d_{pk_H(X)}(X, Y, \ldots, Z, F!pk_H(F))$ is publicly available on the bulletin board after the entire path $X \rightarrow Y \ldots Z \rightarrow F$ has become strong.

Second, as an implementation detail, our actual protocol uses hybrid encryption to emulate the two encryption layers because HIBE cannot be used to encrypt arbitrary messages. We encrypt the message with a symmetric key derived from two freshly generated random strings (both drawn from the domain of HIBE encryption) and encrypt one of the random strings to the key $d_{pk_H(A)}(A \rightarrow^+ C \Rightarrow^+ D!)$ and the other to the key $d_{pk_H(D)}(D \rightarrow^+ E \Rightarrow^* B)$.

Scenario 5: Arbitrary iterations of weak-strong chains. We generalize Scenario 4 to arbitrary iterations of weak chains followed by strong chains. Here, A wants to encrypt to a chain of the form $A = C_0 \rightarrow^+ C'_0 \Rightarrow^+ C_1 \rightarrow^+ C'_1 \Rightarrow^+ \ldots C_n \rightarrow^+ C'_n \Rightarrow^* B$ for an arbitrary $n \geq 0$. Our solution in this case generalizes that of Scenario 4, which was the special case $n = 1$. We use $n+1$ layers of HIBE. The first n layers are to the finalized keys $d_{pk_H(C_i)}(C_i \rightarrow^+ C'_i \Rightarrow^+ C_{i+1}!pk_H(C_{i+1}))$ for $i \in \{0, \ldots, n-1\}$. The last encryption is to the decryption key $d_{pk_H(C_n)}(C_n \rightarrow^+ C'_n \Rightarrow^* B)$, which is owned by B. B can obtain all $n+1$ decryption keys after all links in the chain have been authenticated.

A conceptually simpler solution is to encrypt with two layers independent of n: one to the key $d_{pk_H(C_0)}(C_0 \rightarrow^+ C'_0 \Rightarrow^+ C_1 \rightarrow^+ C'_1 \Rightarrow^+ \ldots C_n!pk_H(C_n))$ and the other to the key $d_{pk_H(C_n)}(C_n \rightarrow^+ C'_n \Rightarrow^* B)$. The advantage of our solution compared to this simpler solution is that it uses HIBE encryption with fewer levels (we break the long chain from A to C_n into n subchains). This allows us to limit the cost of link authentication (see Sections IV and VI).

Scenario 6: Arbitrary chains (most general case). We now explain the most general case of our encryption, where A wants to encrypt to an arbitrary chain. Note that every finite chain from A to B can be written (uniquely) in the form $A \Rightarrow^* C_0 \rightarrow^+ C'_0 \Rightarrow^+ C_1 \rightarrow^+ C'_1 \Rightarrow^+ \ldots C_n \rightarrow^+ C'_n \Rightarrow^* B$ for some n. This form is nearly identical to that of Scenario 5 except that, here, A is not equal to C_0. Instead, there is a strong chain from A to C_0. However, this makes no difference to the protocol. The protocol in Scenario 5 only requires that A possess C_0's public HIBE key, $pk_H(C_0)$, in order to create the first encryption layer to the key $d_{pk_H(C_0)}(C_0 \rightarrow^+ C'_0 \Rightarrow^+ C_1!pk_H(C_1))$. A can obtain that key from the bulletin board as there is a strong chain from A to C_0. Hence, the encryption that A performs in the general case is identical to that in Scenario 5.

Key compromise. Key compromise refers to the adversary obtaining the private keys of an *honest* party. Even though key compromise is outside our threat model, we note that the effect of key compromise in an IWoT is no worse than its effect in a standard WoT. If the adversary compromises an identity, it can authenticate fake chains starting from that identity. Any encryption to a path that includes a node from any such fake chain provides no provable confidentiality. This can be mitigated to some extent by encrypting to several paths simultaneously (see Section VII-A).

IV. Construction Details

We present the cryptographic construction that realizes the IWoT scheme described above. The construction employs a NM-CCA-secure [14] symmetric encryption scheme, an IND-CCA-secure asymmetric (public-key) encryption scheme [29], an EUF-CMA-secure digital signature scheme [18] and an IND-HID-CCA-secure HIBE scheme [17]. Additionally, we assume the existence of a public bulletin board functionality $\mathcal{F}_{BB}$ [32].

A. Initial setup

At the start of an IWoT, it is assumed that no two parties have authenticated each other and the bulletin board is empty. Common parameters for the encryption, signature and HIBE algorithms that can be used by all parties, e.g., curves, generators and such, are created and posted on the bulletin board. Prior to participating in the IWoT, a party N generates its private and public keys for signing ($sk_S(N)$ and $pk_S(N)$) and encryption ($sk_E(N)$ and $pk_E(N)$), a HIBE master secret key ($sk_H(N)$) and a corresponding public key ($pk_H(N)$).

B. Link authentication

To authenticate a party N, a party M verifies N's identity offline and then adds the following to the bulletin board:

1) N's three public keys $pk_S(N)$, $pk_E(N)$ and $pk_H(N)$, signed by M's signing key $sk_S(M)$.
2) For every HIBE decryption key $d_{pk_H(A)}(A, \ldots, M)$ accessible to M and for every HIBE identity $(N, \ldots, Z)$ such that $d_{pk_H(N)}(N, \ldots, Z!pk_H(Z))$ exists on the bulletin board:
   a) The HIBE decryption key $d_{pk_H(A)}(A, \ldots, M, N, \ldots, Z)$ *encrypted* with Z's public key $pk_E(Z)$. (This allows Z and only Z to obtain the decryption key $d_{pk_H(A)}(A, \ldots, M, N, \ldots, Z)$.)
   b) The HIBE decryption key $d_{pk_H(A)}(A, \ldots, M, N, \ldots, Z!pk_H(Z))$ in the clear. (This establishes *publicly* that the entire path $A, \ldots, M, N, \ldots, Z$ is now strong.)

It is an invariant that the key $d_{pk_H(A)}(A, \ldots, Z!pk_H(Z))$ exists on the bulletin board if and only if every party in the chain $A, \ldots, Z$, except Z, has authenticated the next party on the chain.

In general, step (2) can be expensive: In rare, bad cases, M may end up generating $O(n!)$ decryption keys, where n is the number of parties in the IWoT. To prevent this blow up, a practical system may limit the lengths of chains $A, \ldots, M, N, \ldots, Z$ considered to small numbers like 2 or 3. This also means that during encryption (see below) the

maximum length of a weak chain on the chosen path must be small. While this can, in principle, make communication between some pairs of parties impossible, in Section VI, we show through measurements on an actual web of trust that this restriction is reasonable in practice: It bounds authentication cost and does not seriously limit the ability to communicate.

C. Encryption

To encrypt a message m for a recipient B, the sender A picks any sequence of parties starting in A and ending in B that it expects will become strong eventually. It factors this sequence (uniquely) to the form $A \Rightarrow^* C_0 \rightarrow^+ C_0' \Rightarrow^+ C_1 \rightarrow^+ C_1' \Rightarrow^+ \ldots C_n \rightarrow^+ C_n' \Rightarrow^* B$. For best performance, A should pick a sequence that makes each subchain of the form $C_i \rightarrow^+ C_i' \Rightarrow^+ C_{i+1}$ as short as possible. A then generates $n+1$ random symmetric keys, $k_0, \ldots, k_n$ and symmetrically encrypts the message m with a key $k_0 \oplus \ldots \oplus k_n$ derived from these $n+1$ keys. If even an unauthenticated public encryption key, $pk_E(B)$, for B is available, A adds an extra layer of asymmetric encryption with that key. A then HIBE encrypts each k_i for $0 \leq i \leq (n-1)$ to the identity $C_i \rightarrow^+ C_i' \Rightarrow^+ C_{i+1}!pk_H(C_{i+1})$ (to do this, it needs $pk_H(C_i)$, which is available from the bulletin board, since C_i lies at the end of a strong chain). A also HIBE encrypts k_n to the identity $C_n \rightarrow^+ C_n' \Rightarrow^* B$. It then sends the ciphertext of m, and the ciphertexts of the $n+1$ keys $k_0, \ldots, k_n$ to B over any channel (possibly over the bulletin board).

D. Decryption

The recipient B attempts to decrypt a ciphertext encrypted to the path $A \Rightarrow^* C_0 \rightarrow^+ C_0' \Rightarrow^+ C_1 \rightarrow^+ C_1' \Rightarrow^+ \ldots C_n \rightarrow^+ C_n' \Rightarrow^* B$ only after all chains of the form $C_i \rightarrow^+ C_i'$ have become strong. At this point, it uses the key $d_{pk_H(C_i)}(C_i \rightarrow^+ C_i' \Rightarrow^+ C_{i+1}!pk_H(C_{i+1}))$ from the bulletin board to decrypt the ciphertext of k_i and obtain k_i for $0 \leq i \leq (n-1)$. It uses the key $d_{pk_H(C_n)}(C_n \rightarrow^+ C_n' \Rightarrow^* B)$, which it can also obtain from the bulletin board, to decrypt the ciphertext of k_n and get k_n. This allows it to compute $k_0 \oplus \ldots \oplus k_n$ and decrypt the ciphertext of the message m.

E. Example

This section presents an example that illustrates our construction. Consider the relationship graph in Figure 3. The only pre-existing strong path is $C \Rightarrow F$, so F's public keys, a signature on those by C, the key $d_{pk_H(C)}(C \Rightarrow F!pk_H(F))$ and the ciphertext $Enc(pk_E(F), d_{pk_H(C)}(C \Rightarrow F))$ are the only things on the bulletin board initially.

A now wants to encrypt a message m for B. For that, A has to choose a path to B. A could pick one of several paths. F is on all such paths, so it has to be included. Utilizing the already strong link between C and F allows A to use two subchains. The second subchain should be as short as possible (to minimize the number of key escrows), so using $E \rightarrow G$ should be avoided. Other than this, it does not matter whether $F \rightarrow G \rightarrow B$ or $F \rightarrow E \rightarrow B$ is chosen as the second subchain.

Suppose the complete chain chosen by A is $A \rightarrow D \rightarrow C \Rightarrow F \rightarrow G \rightarrow B$, and there is no public key known for B. A now generates two keys k_1 and k_2 and computes $k = k_1 \oplus k_2$.

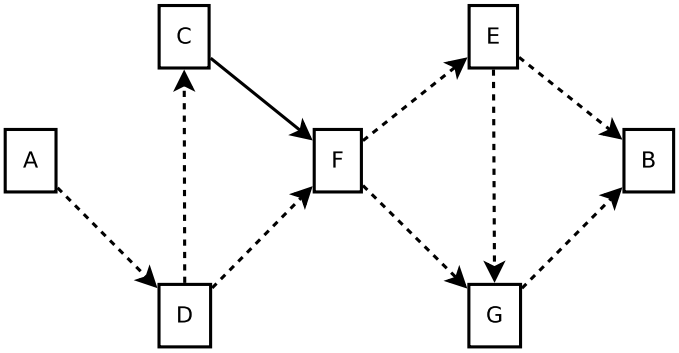

Fig. 3. Sample relationship graph for the example of Section IV-E. Dotted lines represent weak links and full lines represent strong links.

Also, it retrieves $pk_H(F)$ from the bulletin board. A builds the ciphertext as follows:

$$\begin{aligned} c \;=\; & Enc_S(k, m), \\ & Enc_H(pk_H(A), (A, D, C, F!pk_H(F)), k_1), \\ & (A, D, C, F), \\ & Enc_H(pk_H(F), (F, G, B), k_2), \\ & (F, G, B) \end{aligned}$$

This ciphertext is then sent from A to B on any channel. For B to be able to decrypt the message, the weak links on the chain must be authenticated in some order. Suppose D authenticates to C first, so $D \rightarrow C$ is the first link to become strong. For this, D checks C's identity and then D creates signatures on C's public keys with its own private signature key $sk_S(D)$ and posts the signatures on the bulletin board. Additionally, D generates several HIBE keys. Since $C \Rightarrow F!pk_H(F)$ is on the bulletin board, the following additional material is posted to the bulletin board by D:

$$\begin{aligned} & d_{pk_H(D)}(D, C!pk_C) \\ & d_{pk_H(D)}(D, C, F!pk_F) \\ & Enc_P(pk_E(C), d_{pk_H(D)}(D, C)) \\ & Enc_P(pk_E(F), d_{pk_H(D)}(D, C, F)) \end{aligned}$$

From now on, the link from D to C is strong, written $D \Rightarrow C$. Afterwards, the remaining weak links ($A \rightarrow D$, $F \rightarrow G$ and $G \rightarrow B$) authenticate. Once all links have been authenticated, the following two materials exist on the bulletin board (among other things).

$$\begin{aligned} & d_{pk_H(A)}(A, D, C, F!pk_H(F)) \\ & Enc_P(pk_E(B), d_{pk_H(F)}(F, G, B)) \end{aligned}$$

B uses the second component of the ciphertext c with the first key above to obtain k_1. From the second ciphertext above, B obtains $d_{pk_H(F)}(F, G, B)$. Combining this with the fourth component of the ciphertext c, it obtains k_2. From this, it computes $k = k_1 \oplus k_2$ and, then, from the first component of the ciphertext, it obtains the original message m.

V. Security Analysis

In this section, we formally define the security of our scheme as an ideal functionality. We then define a simulator that intervenes between the functionality and the adversary, and prove the security of the protocol via a simulation.

Initialize. Initialize with a set of identities $\mathcal{ID}$ and a set of parties $\mathcal{P}$.

Registration. Any party can, at any point in time, decide to register for their identity in $\mathcal{ID}$. Upon receiving a message $(\texttt{Reg}, R)$ from party P_r, do the following:

1) Hand $(\texttt{Reg}, \mathsf{P}_r, R)$ to the adversary $\mathcal{S}$.
2) return the adversary's response $Success$ or $Failure$ to P_r.

Link Authentication. Upon receiving a message $(\texttt{Auth}, R, \langle \mathsf{P}_j, S \rangle)$ from party P_r, do the following:

1) Hand $(\texttt{Auth}, \langle \mathsf{P}_r, R \rangle, \langle \mathsf{P}_s, S \rangle)$ to the adversary $\mathcal{S}$
2) return the adversary's response $Success$ or $Failure$ to P_r.

Encryption. Upon receiving a message $(\texttt{Encrypt}, m, \mathcal{C})$, from party P_r, do as follows:

1) Give $(\texttt{Encrypt}, |m|, \mathcal{C})$ to the adversary $\mathcal{S}$.
2) Receive c from the adversary $\mathcal{S}$ and store $(m, c, \mathcal{C})$ in dec.
3) Return c to P_r.

Decryption. Upon receiving a message $(\texttt{Decrypt}, c)$, from party P_r, do the following:

1) Check if the following conditions hold:
   - There exists entry $(m, c, \mathcal{C}) \in dec$ for c
   - If receiver (or decryptor) party P_e is included in $\mathcal{C}$, then $\mathsf{P}_r = \mathsf{P}_e$, else P_r has to be registered for some identity on the last subchain $\mathcal{SC}_n$.
   - For all subchains $\mathcal{SC}_i \in \mathcal{C}$ (except the last subchain $\mathcal{SC}_n$), check if the last party (and identity) in $\mathcal{SC}_i$ equals the first party (and identity) in $\mathcal{SC}_{i+1}$, i.e. $\mathsf{P}_{e,i} = \mathsf{P}_{b,i+1}$ and $I_{e,i} = I_{b,i+1}$
   - For all possible parties fitting the identities and parties given in the $\mathcal{C}$, check *per subchain from left to right* until one conditions fail or all of them are fulfilled. In particular, for any edge (I, J) in the $\mathcal{C}$ the following hold:
     - There exists a party P_i such that $\langle \mathsf{P}_i, I \rangle$ is registered as suggested by $\mathcal{S}$
     - There exists a party P_j such that $\langle \mathsf{P}_j, J \rangle$ is registered as suggested by $\mathcal{S}$
     - A $strength$ query for $(\langle \mathsf{P}_i, I \rangle, \langle \mathsf{P}_j, J \rangle)$ to $\mathcal{S}$ returns strong.
     - For any first element of any subchain $\mathcal{SC}_a$ (for $a \in \{1, \ldots, n\}$), this $\mathsf{P}_i = \mathsf{P}_{b,a}$
     - For any last element of any subchain $\mathcal{SC}_a$ (for $a \in \{1, \ldots, n\}$), this $\mathsf{P}_j = \mathsf{P}_{e,a}$
2) If all conditions satisfy, then hand $(\texttt{ValidDecrypt}, m, c)$ to P_r and $(\texttt{ValidDecrypt}, c)$ to $\mathcal{S}$, else hand $(\texttt{InvalidDecrypt}, c)$ to both.

Fig. 4. An ideal functionality ($\mathcal{F}_{IWoT}$) for the identity web of trust

A. Security Definition

We formally define the security goals of our design using the ideal/real world paradigm.

Setup and Assumptions. Our ideal functionality ($\mathcal{F}_{IWoT}$) expects a set of identities $\mathcal{ID} = \{A, B, \ldots Z\}$, a set parties $\mathcal{P} = \{\mathsf{P}_1, \mathsf{P}_2, \ldots \mathsf{P}_\eta\}$ and an adversary $\mathcal{S}$. Let $\mathcal{P}_{hon}$ be the set of honest parties, and $\mathcal{P}_{mal}$ be the set of malicious (impersonators) parties such that these sets partition $\mathcal{P}$. Honest parties are bound to use "their" actual identity, while malicious parties may claim any polynomially bounded number of identities. The number of malicious parties is also assumed to be bounded polynomially in the security parameter. Malicious parties may collude with each other.

Let variable $\mathcal{C}$ denote an identity chain that consists of subchains, where each subchain $\mathcal{SC}_i$ is of the type $\mathcal{SC} = (\mathcal{P}, \mathcal{ID}, \mathcal{ID}^+, \mathcal{P}^{\{0,1\}})$. The last subchain need not necessarily have a party at its end in case the recipient is not yet registered. There may be any number of strong links on a subchain. Let $\mathsf{P}_{b,i}$ and $I_{b,i}$ denote the beginning party and identity of the beginning party of the i-th subchain and $\mathsf{P}_{e,i}$ and $I_{e,i}$ denote the ending party and identity of the ending party of the i-th subchain. The last subchain is $\mathcal{SC}_n$.

Message Flow. We expect the ideal functionality to access the bulletin board functionality $\mathcal{F}_{BB}$ through the adversary $\mathcal{S}$, who should transform cryptographic information from $\mathcal{F}_{BB}$ to non-cryptographic information required by the functionality, and vice-versa. In this regard, $\mathcal{F}_{IWoT}$ makes two kind of queries to the adversary: a $\mathcal{RP}$ query to check if a party is registered with an identity, and a $strength$ query to check if a link from a party (and her identity) to another is strong or weak.

The $\mathcal{F}_{IWoT}$ functionality (Figure 4) allows the following four messages/actions by any party: $Registration$, $Link\ Authentication$, $Encryption$ and $Decryption$.

As the adversary may impersonate anyone, the ideal functionality allows parties to register to any identities using the $Registration$ function. When any party registers an identity, this pairing is passed on to the adversary $\mathcal{S}$ (to maintain it in $\mathcal{F}_{BB}$) such that $\mathcal{F}_{IWoT}$ can check the pairing later. Notice that the $Registration$ step is completely optional, but allows any party to register for the identity they committed to in order to improve security. Honest nodes will only register for one identity in the whole functionality, while malicious

(impersonator) parties can register for any number of identities.

A $Link\ authentication$ message is used by a party to strengthen links (strong links represent authentication in the "real world"). It takes two parties, one authenticating and one authenticated, along with their suggested identities and passes this information to the adversary. The effect of a successful authentication is twofold: Subsequent $strength$ queries on the authenticated link will answer $strong$ in place of $weak$, and $\mathcal{RP}$ queries for both identity-party pairs return affirmatively in the future.

For the $Encryption$ message, the functionality requires a plaintext message and an identity chain from the submitting party. This chain may include the recipient party P_e if it is known to the encryptor. The functionality returns a ciphertext symbol that is provided by the adversary and can later be used for decryption. To maintain the confidentiality of the messages, only the message length and the chain $\mathcal{C}$ is conveyed to the adversary $\mathcal{S}$.

For the $Decryption$ message, the functionality requires a ciphertext c from the submitting party P_r. The submitting party must be on the last subchain in the corresponding identity chain (this allows representing HIBE decryption of the message by anyone on the last HIBE subchain, as in the last mile problem in the real world). If the chain $\mathcal{C}$ contains the recipient party P_e, the submitting party must be the recipient. The functionality starts by verifying the sanity of the chain $\mathcal{C}$ by checking if all consecutive subchains are consistent with each other. Afterwards, it checks if there is one combination of parties associated with the given identities in the chain, such that all subsequent identity pairs in the chain have $strong$ connections. If it does not find such a combination, the functionality returns `InvalidDecrypt`. If the functionality finds a combination such that all the above conditions hold, the original message is returned to P_r.

Definition 1. *An IWoT protocol is simulation secure in the $(\mathcal{F}_{BB})$-hybrid model if for all PPT adversaries $\mathcal{A}$ in the real world who actively infiltrate the protocol by posting corrupt identities and use these to create corrupt links in the protocol, there exists a simulator $\mathcal{S}$ in the ideal world execution with $\mathcal{F}_{IWoT}$, which corrupts the same parties and produces an output identically distributed to the output of $\mathcal{A}$ in the real world.*

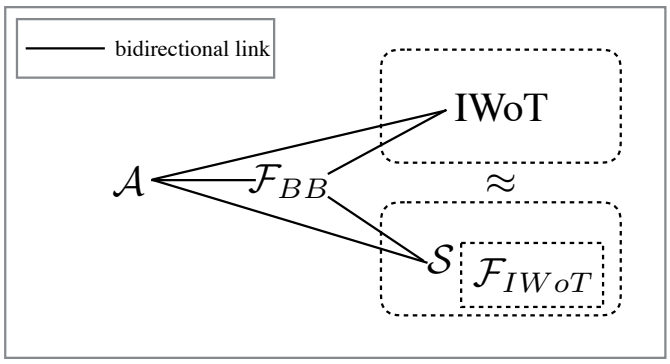

Fig. 5. Overview of the security analysis set-up

B. Security Proof

Figure 5 shows a graphical sketch of our security analysis framework. As previously mentioned, the adversary can create an arbitrary number of corrupt identities (including ones for the sender and the receiver) and links between those, but we expect that no honest party in the network would create a link to them. It is also not allowed to corrupt existing honest identities (i.e., we do not allow for key compromise).

Theorem 2 (Simulation Security)**.** *If Enc_H is an IND-HID-CCA secure HIBE scheme, Enc_E is an IND-CCA secure PKE scheme and Enc_S is a NM-CCA-secure symmetric encryption scheme, Sig_S is an EUF-CMA signature scheme, and assuming that every user has access to the current social graph, then the IWoT protocol is simulation secure as formalized in Definition 1.*

Proof Outline: We provide a simulator $\mathcal{S}$ (Figure 6) for the ideal world execution with $\mathcal{F}_{IWoT}$ in the hybrid model with the bulletin board functionality $\mathcal{F}_{BB}$. An informal description of the simulator is as follows.

$\mathcal{S}$ runs on behalf of the good parties, and simulates the ideal world such that the adversary with its malicious parties cannot distinguish between the ideal and the real world.

The simulator runs the key generation algorithm on behalf of the honest parties and, thus, knows all public as well as private keys for the honest parties. It also knows all (relevant) public keys for the fake parties from $\mathcal{F}_{BB}$. This makes the registration as well as the authentication processes very simple for $\mathcal{S}$ as it can perform all necessary authentication operations on behalf of the honest parties accurately and in a manner that is indistinguishable from the real world. The simulator $\mathcal{S}$ locally maintains a map between $\mathcal{F}_{IWoT}$ parties P_r and pk_r published on $\mathcal{F}_{BB}$ for some identity R. This mapping is efficient.

The simulator's key challenge comes during the encryption calls. Although $\mathcal{F}_{IWoT}$ tells it the identity chain $\mathcal{C}$ used for the ciphertext as well as the length of the plaintext, it does not know the plaintext. The simulator $\mathcal{S}$ instead encrypts $0^{|m|}$ for the same chain. This is sufficient since it is impossible to distinguish the encryptions of $0^{|m|}$ and m by comparing ciphertexts (we employ IND-secure encryption primitives). Note that the adversary will be able to decrypt some intermediate HIBE encryptions with the public finalized keys; however, we employ those as the key encapsulation mechanism (KEM) since the decrypted message (symmetric key share) will be indistinguishable from uniform randomness in the real as well as the ideal world. The trick also works if an honest sender ends up choosing some fake parties on its path to an honest receiver as such chains never get completely strong and the encrypted messages will never get decrypted. In this case, even if the fake party belongs to the last mile and can derive the private keys for the receiver in the last subchain, thanks again to our use of KEM, the decrypted symmetric key share will be indistinguishable from uniform randomness.

Although $\mathcal{S}$ will not provide similar security for the completely adversarial identity chain (the adversary can fully decrypt and distinguish between m and $0^{|m|}$ for such a chain), it does not have to—the simulator is only responsible for

The simulator is provided with the set of identities $\mathcal{ID}$ and a set of parties $\mathcal{P}$. Let $\mathcal{P}_{hon}$ and $\mathcal{P}_{mal}$ respectively denote honest and malicious parties, and the simulator knows both of these sets.

During party registration. Upon receiving $(\texttt{Reg}, P_r, R)$ from the functionality:

1) Map P_r to the HIBE, PKE, and signature key set pk_r.
2) Check if the (pk_r, R) are already registered on $\mathcal{F}_{BB}$.
3) If yes, return $Failure$
4) Otherwise, run the key generation algorithm for P_r to generate public-private keys for HIBE, PKE and signature, create a map entry for P_r and pk_r locally, post those (i.e. pk_r) with identity R on $\mathcal{F}_{BB}$, and return $Success$.

During authentication request. Upon receiving $(\texttt{Auth}, \langle P_r, R\rangle, \langle P_s, S\rangle)$ from the functionality:

1) Generate and register the HIBE, PKE, signature keys for P_r and/or P_s, as above, if they are not yet registered.
2) Authenticate public keys associated with (P_s, S) from the signing key sk_R for (P_r, R).
3) Extend the HIBE key for all HIBE for which P_r has private keys including its own HIBE. All necessary information is available from $\mathcal{F}_{BB}$.

During encryption. Upon receiving $(\texttt{Encrypt}, |m|, \mathcal{C})$ from the functionality:

1) Encrypt $0^{|m|}$ (as an encryption of "garbage") for the chain $\mathcal{C}$. Notice that the simulator knows (locally or through $\mathcal{F}_{BB}$) all necessary public keys to encrypt $0^{|m|}$ correctly.
2) Return the ciphertext c to the functionality.

$\mathcal{RP}$ **and** $strength$ **queries.** Upon receiving the $\mathcal{RP}$ and $strength$ queries from the functionality:

1) Replace the functionality parties with pubic key sets.
2) Run the query correctly with $\mathcal{F}_{BB}$ and send the answer back to the functionality.

Fig. 6. The simulator S for the ideal functionality

honest parties, and does not need to simulate an instance involving no honest parties.

The simulator $\mathcal{S}$ does not need to do any operation during a decryption call as the ideal functionality performs all the necessary work.

This shows that the output distribution of the simulator $\mathcal{S}$ is identical to the output distribution of the real IWoT protocol's distribution. It is also easy to see that $\mathcal{S}$ remains as efficient as all honest parties together.

Therefore, the IWoT protocol is simulation secure as formalized in Definition 1. ■

We can easily extend our analysis to allow the adversary to act in an honest but curious way, i.e., it can create an identity for itself, but is then not allowed to verify other dishonest identities using this identity. However, to keep the exposition simple, we do not consider such an adversary in the current model.

VI. Implementation and Performance Analysis

We have built a library that implements all the functionality of IWoTs, including authentication, encryption and decryption. This library is not yet optimized, but it suffices to conservatively estimate how IWoTs would perform in practice. Our library is written in Java 1.7. We use Kalium, a Java binding for NaCl [6] by Bernstein, Lange and Schwabe for non-pairing-based cryptography, and jPBC [13], a Java binding for the PBC library by Lynn [22] for identity-based cryptography. NaCl uses Ed25519 for EC signing, a combination of Curve25519, Salsa20, and Poly1305 is used for asymmetric encryption and for the identity-based encryption, a type-F curve with 384 random bits is used. Our HIBE implementation is based on a paper by Gentry and Silverberg [17]. Our library contains a small interactive command line interface for experimentation and performance evaluation.

All experimental numbers shown here were obtained on a 2x Intel Xeon CPU E5-2667 v2 with 16 physical and 16 logical cores, 256GB RAM, running Debian 8 "Jessie". For storage, we used two 1TB HDDs in a RAID-1 configuration.

Social graph. To conduct experiments, we needed a social graph. For this, we used the publicly available graph from the SKS OpenPGP key server [24]. This graph lists nodes (public keys) and edges (which key verifies which other key), along with time stamps when each edge came into existence. This allows us to reconstruct snapshots of the graph over time. Figure 7 shows some basic statistics on graph snapshots every 3 years from 1999 to 2014 (we removed nodes that had no adjacent edges; these are not counted in the graph). The graph also shows the size of the largest connected cluster of nodes in each year. Depending on the year, this varies from 10% to 23% of the total size of the graph, indicating that in each year, most of the graph is not densely connected.

For measuring the performance of IWoTs, we treat each node as a separate party (and separate identity). PGP has no notion of weak edges. Consequently, for the purpose of our experimentation, we assume that any edge in the PGP graph is weak for 6 months prior to its addition to the graph, at which point it becomes strong.

Authentication. Authentication is an expensive (but infrequent) operation in IWoTs because when M authenticates N,

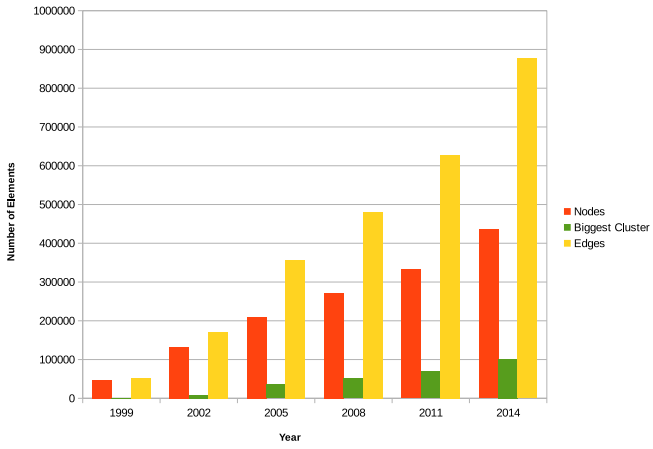

Fig. 7. Numbers of nodes and edges in our graph in each year

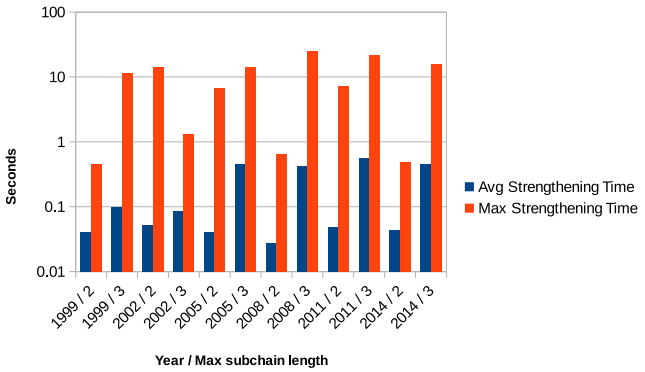

Fig. 8. Average and maximum time for authentication under maximum subchain lengths of 2 and 3

for every key $d_{pk_H(A)}(A, \ldots, M)$ accessible to M and every key $d_{pk_H(N)}(N, \ldots, Z!pk_H(Z))$ on the bulletin board, N must generate two new keys. In the worst-case, with n nodes, N may have to generate $O(n!)$ keys, which is intractable. In practice, however, the length of the sequence $A, \ldots, M, N, \ldots, Z$ can be limited to a small constant k, by limiting the maximum length of weak subchains chosen during encryption to k. Figures 8 and 9 show the maximum and average time taken and the maximum and average number of keys generated during an authentication operation (the maximum and average are taken over 1,000 randomly chosen authentication operations in each year). Numbers are shown for both $k = 2$ and $k = 3$. As can be seen, even with a maximum chain length of $k = 3$, the maximum time taken to authenticate is 14.5 seconds in our experiments, which is reasonable since authentication is relatively infrequent. The average time does not exceed 1s in any year. (Note, however, that this measurement is made on server-grade CPUs. Smaller CPUs may require more time. Nonetheless, time of the order of seconds or even a few minutes for each authentication looks reasonable.)

Total key storage. Another potentially significant overhead is that of key storage, since a node may have to store many HIBE decryption keys. Specifically, for every strong path of the form $A, \ldots, N$, N must store the key $d_{pk_H(A)}(A, \ldots, N)$.

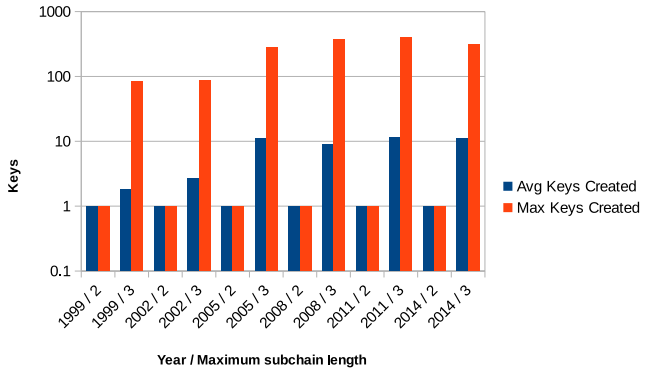

Fig. 9. Average and maximum numbers of created keys under maximum subchain lengths of 2 and 3

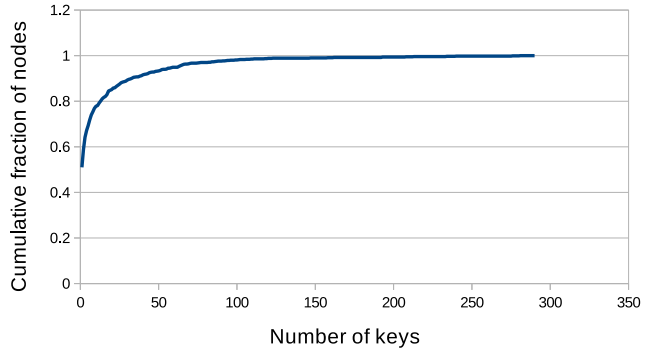

Fig. 10. Cumulative distribution of keys to be stored at each node for a random sample size of 1000 nodes for 2014 and maximum subchain length of 3

Figure 10 shows the cumulative distribution of the number of keys that a node must store (over a sample of 1000 nodes in the year 2014 for $k = 3$). Most nodes need to store no more than 100 keys, which is very manageable, given that each key is no more than a few kilobytes in size (including metadata).

Feasibility of message encryption under subchain length limits. If we limit the maximum length of unauthenticated chains during encryption, an obvious question is how many pairs of nodes can communicate. To answer this question, we implemented an algorithm to find paths between any given pair of nodes, while minimizing the length of the longest unauthenticated subchain. Figure 11 shows the percentage of node pairs between which messages can be encrypted with different upper-bounds on subchain lengths in each year. With a bound of $k = 3$, at least 97% of node pairs can communicate in every year, which suggests that $k = 3$ is a reasonable limit.

Encryption and decryption times. In our final experiment, we measure the amount of time it takes to encrypt and decrypt a message between two randomly chosen nodes, where paths are selected to minimize the length of the longest unauthenticated subchain. Figure 12 shows the maximum and average encryption and decryption times over 1000 randomly chosen node pairs for $k = 2, 3, 4$, across years. As can be seen, in all cases, the average times for both encryption and decryption are under 1s, which is very reasonable. The maximum times

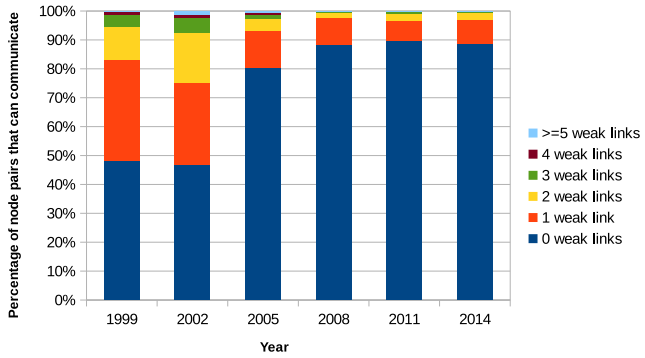

Fig. 11. Percentage of node pairs that can communicate under different weak subchain length restrictions

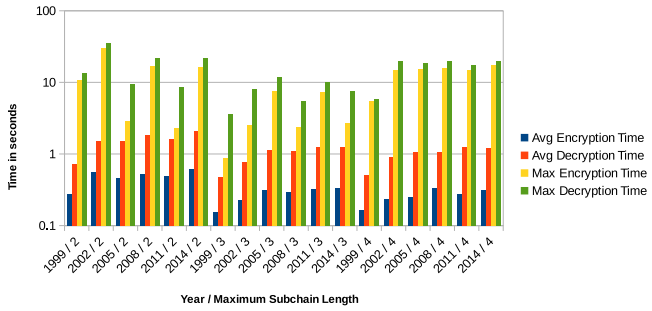

Fig. 12. Average and maximum encryption and decryption time under different weak subchain maximum lengths

are of the order of tens of seconds, which is also reasonable.

VII. Enhancements

This section describes three enhancements to the basic IWoT scheme.

A. Use of Multiple Paths for Encryption

In the basic IWoT scheme, a sender encrypts a message binding to a single path from itself to the recipient. The security of such encryption relies on the assumption that each identity on the path authenticates the next identity correctly. To weaken this assumption, the basic IWoT scheme can be extended by binding the encryption to more than one path simultaneously. To do this, ciphers corresponding to additional paths are added as layers to the message's ciphertext. The property attained is that the message can be decrypted only when all paths have become strong and, hence, that only the intended recipient can decrypt the message if all identities along *any one path* perform authentication correctly. This is similar to the use of multiple paths to authenticate a public key in a standard web of trust.

The use of multiple paths is also attractive for another reason: It reduces the exploitability of the last mile problem (Section III-B)—to exploit the key escrows, the attacker must compromise a key escrow at the tail of *every* path to which the message's encryption is bound.

B. Source Authentication

The basic IWoT scheme guarantees to the sender that only the intended recipient can obtain the sent message. However, it does not provide source authentication, i.e., it does not guarantee to the recipient that the message was actually sent by the claimed sender. Source authentication is easily added to the scheme by having the source sign the message with its signing key and making link authentication bi-directional, meaning that when M authenticates to N, N also authenticates to M (and M signs N's public keys). When the recipient decrypts a message, a strong chain must exist from the sender to the recipient, and because of the bi-directional authentication at each link, the recipient can authenticate the sender's verification key. Assuming that the recipient trusts that all identities on the path authenticate correctly, she can be certain that the claimed sender actually sent the message.

C. Ciphertext anonymity

In the IWoT protocol presented so far, anybody can determine the receiver as well as the whole identity chain for a ciphertext by observing the ciphertext. It is, however, possible to modify the IWoT protocol such that it achieves *ciphertext anonymity*, i.e, an adversary cannot identify the recipient and the complete identity chain from the ciphertext [5].

To attain ciphertext anonymity, we not only need to modify the ciphertext format for IWoT, but also replace the employed HIBE scheme; the currently employed Gentry-Silverberg HIBE scheme [17] leaks information about the HIBE-depth of the recipient on the last subchain. Instead, we could employ a newer HIBE scheme [27] that provides ciphertext anonymity for ciphertexts.

Currently, ciphertexts contain the associated identity chains in plain text (lines 3 and 5 of the ciphertext c in the example of Section IV-E). We cannot remove all of these identities from the ciphertext as otherwise the complexity of decryption for the receiver will increase significantly. Instead, we could remove only the last subchain, and expect the receiver to try decrypting the ciphertext with every new HIBE decryption key she receives. We can employ *another layer* of symmetric encryption with a new key k_{out} to hide the identities in all but the last subchain and HIBE-encrypt k_{out} with the last subchain. The other subchains' HIBE encryptions and plaintext identities are also re-encrypted with k_{out} as otherwise any curious observers can determine subchain identities by continuously trying decryptions of those HIBE encryptions with the published finalized keys.

As an illustration, the ciphertext c from the example in Section IV-E would be transformed to the following anonymous ciphertext, c_{anon}:

$$\begin{aligned} c_{anon} \quad = \quad & Enc_S(k, m), \\ & Enc_S(k_{out}, (Enc'_H(pk_H(A), \\ & \qquad (A, D, C, F!pk_H(F)), k), \\ & \qquad (A, D, C, F)), \\ & Enc'_H(pk_H(F), (F, G, B), k_{out}) \end{aligned}$$

Here, Enc'_H is a ciphertext-anonymous HIBE scheme [27]. To obtain the symmetric decryption key k to obtain the message m the receiver will have to obtain k_{out} using the HIBE decryption key for the last subchain (F, G, B), then use k_{out} to decrypt the

outer layer of the layered encryption to obtain the remaining subchain (A, D, C, F), and then obtain the HIBE finalized key for $(A, D, C, F!pk_H(F))$ from the bulletin board.

It is possible to hide the lengths of the identity subchains except the last one by attaching 'dummy' identities to the chains to pad their representations to a fixed length. Additionally, the ciphertext-anonymous HIBE construction does not leak anything about the last subchain to any recipient of the message. Thus, the intended recipient has to try decrypting the final part of the message with every non-finalized HIBE key they receive. This price is conceptually unavoidable for an anonymous ciphertext.

VIII. Summary

IWoT re-designs WoTs (and other PKIs by extension) to use hierarchical identity-based encryption (HIBE) in addition to public key cryptography. This novel construction allows a sender to encrypt and transmit a message even before a chain of trust to the recipient has been established. IWoTs are particularly useful for ad hoc, uncoordinated communication in presence of powerful adversaries and censors, who can both compromise key authorities (so conventional HIBE is ineffective) and harm the sender (so waiting for a chain of trust to be established, as would be forced in a WoT, is not always a prudent option). We have shown that our construction is secure and efficient enough to be used in a practical system.

Acknowledgement. This work is supported by the Indo-German Max Planck Center for Computer Science (IMPECS).

References

[1] "Google?s Gmail Hacked This Weekend? Tips To Beef Up Your Security," http://www.huffingtonpost.com/paul-everton/googles-gmail-hacked-this_b_3641842.html.

[2] "Web of Trust Info," https://github.com/WebOfTrustInfo, 2016.

[3] A. Abdul-Rahman, "The PGP trust model," in *EDI-Forum: the Journal of Electronic Commerce*, vol. 10, no. 3, 1997, pp. 27–31.

[4] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Advances in Cryptology—ASIACRYPT 2003*, 2003, pp. 452–473.

[5] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval, "Key-privacy in public-key encryption," in *Advances in Cryptology—ASIACRYPT 2001*. Springer, 2001, pp. 566–582.

[6] D. J. Bernstein, T. Lange, and P. Schwabe, "The security impact of a new cryptographic library," in *Progress in Cryptology–LATINCRYPT 2012*. Springer, 2012, pp. 159–176.

[7] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *Advances in Cryptology—EUROCRYPT 2005*. Springer, 2005, pp. 440–456.

[8] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Advances in Cryptology—CRYPTO 2001*. Springer, 2001, pp. 213–229.

[9] N. Borisov, I. Goldberg, and E. A. Brewer, "Off-the-record communication, or, why not to use PGP," in *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society, WPES 2004, Washington, DC, USA, October 28, 2004*, 2004, pp. 77–84.

[10] J. Clark and P. C. van Oorschot, "SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements," in *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013, pp. 511–525.

[11] CNet, "Feds put heat on Web firms for master encryption keys," http://www.cnet.com/news/feds-put-heat-on-web-firms-for-master-encryption-keys/, 2013.

[12] C. Cocks, "An identity based encryption scheme based on quadratic residues," in *Cryptography and coding*. Springer, 2001, pp. 360–363.

[13] A. De Caro and V. Iovino, "jPBC: Java pairing based cryptography," in *Computers and Communications (ISCC), 2011 IEEE Symposium on*. IEEE, 2011, pp. 850–855.

[14] D. Dolev, C. Dwork, and M. Naor, "Nonmalleable Cryptography," *SIAM J. Comput.*, vol. 30, no. 2, pp. 391–437, jan 2000. [Online]. Available: http://dx.doi.org/10.1137/S0097539795291562

[15] L. Foundation, "Hyperledger ? Blockchain Technologies for Business," https://www.hyperledger.org/, accessed July 2017.

[16] T. Frosch, C. Mainka, C. Bader, F. Bergsma, J. Schwenk, and T. Holz, "How Secure is TextSecure?" Cryptology ePrint Archive Report 2014/904, 2014.[Online]. Available: https://eprint. iacr. org/2014/904, Tech. Rep.

[17] C. Gentry and A. Silverberg, "Hierarchical ID-based cryptography," in *Advances in cryptologyASIACRYPT 2002*. Springer, 2002, pp. 548–566.

[18] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal on Computing*, vol. 17, no. 2, pp. 281–308, 1988.

[19] G. Hanaoka, T. Nishioka, Y. Zheng, and H. Imai, "An efficient hierarchical identity-based key-sharing method resistant against collusion-attacks," in *Advances in Cryptology—ASIACRYPT99*. Springer, 1999, pp. 348–362.

[20] J. Horwitz and B. Lynn, "Toward hierarchical identity-based encryption," in *Advances in Cryptology—EUROCRYPT 2002*. Springer, 2002, pp. 466–481.

[21] Y. Lindell, "How to simulate it - a tutorial on the simulation proof technique," Cryptology ePrint Archive, Report 2016/046, 2016, http://eprint.iacr.org/2016/046.

[22] B. Lynn, "The pairing-based cryptography (PBC) library," https://crypto.stanford.edu/pbc/, 2010.

[23] M. S. Melara, A. Blankstein, J. Bonneau, E. W. Felten, and M. J. Freedman, "Coniks: Bringing key transparency to end users," in *USENIX Security Symposium 2015*, 2015, pp. 383–398.

[24] Y. Minsky, "The SKS OpenPGP key server," https://sks-keyservers.net/.

[25] P. Mittal, M. Caesar, and N. Borisov, "X-Vine: Secure and Pseudonymous Routing in DHTs Using Social Networks," in *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*, 2012.

[26] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Technical report, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[27] S. C. Ramanna and P. Sarkar, "Efficient (anonymous) compact hibe from standard assumptions," in *Provable Security*. Springer, 2014, pp. 243–258.

[28] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in cryptology*. Springer, 1985, pp. 47–53.

[29] V. Shoup, "OAEP reconsidered," in *Advances in Cryptology—CRYPTO 2001*. Springer, 2001, pp. 239–259.

[30] T. K. Srikanth and S. Toueg, "Simulating authenticated broadcasts to derive simple fault-tolerant algorithms," *Distributed Computing*, vol. 2, no. 2, pp. 80–94, 1987.

[31] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith, "SoK: Secure Messaging," in *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, 2015, pp. 232–249.

[32] D. Wikström, "A Universally Composable Mix-Net," in *TCC*, 2004, pp. 317–335.