A Distributed Private-Key Generator for Identity-Based Cryptography

Aniket Kate Ian Goldberg

David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, Canada N2L 3G1
{akate,iang}@cs.uwaterloo.ca

Abstract

Identity-based cryptography can greatly reduce the complexity of sending encrypted messages over the Internet. However, it necessarily requires a private-key generator (PKG), which can create private keys for clients, and so can passively eavesdrop on all encrypted communications. Although a distributed private-key generator has been suggested as a way to mitigate this problem, to date there have been no practical implementations provided for one. This paper presents the first realistic architecture and an implementation for a distributed private-key generator for use over the Internet.

We improve the adversary model in the proactive verifiable secret sharing scheme by Herzberg et al. and define master-key modification and secret share recovery protocols in our new model. Our periodic master-key modification achieves forward secrecy of the master key; this feature has been missing in other proactive security schemes, but is of great importance in identity-based applications. Recognizing the utility of modifying the set of nodes and the security threshold in a distributed PKG, we present protocols for these operations. We also compare our architecture to other verifiable secret sharing architectures for the Internet and demonstrate that ours has both better message efficiency as well as a more complete feature set. Finally, with a geographically distributed installation of our application, we verify its efficiency and practicality.

Keywords: identity-based cryptography, private-key generator, proactive verifiable secret sharing, distributed key generation

1 Introduction

Identity-Based Cryptography. In 1984, Shamir [42] introduced the notion of identity-based cryptography (IBC) as an approach to simplify public-key and certificate management in a public-key infrastructure (PKI) and presented an open problem to provide a identity-based encryption (IBE) scheme. After seventeen years, Boneh and Franklin [6] proposed the first practical and secure IBE scheme (BF-IBE) using bilinear maps. After this seminal work, in the last few years, significant progress has been made in IBC in the forms of hierarchical IBE [5, 24, 45], identity-based signature (IBS) schemes [11, 46], identity-based authentication and key agreement [12, 40], and other identity-based primitives.

In an IBC system, a client chooses any arbitrary string such as her e-mail address to be her public key. Consequently, with a standardized public-key string format, an IBC scheme completely eliminates the need for public-key certificates. As an example, in an IBE scheme, a sender can encrypt a message for a receiver knowing just the identity of the receiver and importantly, without obtaining and verifying the receiver's public-key certificate. Naturally, in such a system, a client herself is not capable of generating a private key for her identity and there is a trusted party called a *private-key generator* (PKG) which performs the system setup and provides private keys to system clients. In a practical IBE scheme, given a client's identity $\texttt{ID}$, the PKG uses a secret called `master-key` to generate the client's private key $d_{\texttt{ID}}$. As the PKG computes a private key for a client, it can decrypt all her messages passively.

This inherent *key escrow* property asks for complete trust in the PKG, which is difficult to find in most realistic scenarios.

Need for the Distributed PKG. The amount of trust placed in the holder of an IBC `master-key` is far greater than that placed in the holder of the private key of a certifying authority (CA) in a PKI. In a PKI, in order to attack a client, the CA has to actively generate a fake certificate for the client containing a fake public-key/private-key pair. In this case, it is often possible for the client to detect and prove the malicious nature of the CA. The CA cannot perform any passive attack; specifically, it cannot decrypt a message encrypted for the client using a client-generated public key and it cannot sign some document for the client, if the verifier gets a correct certificate from the client. On the other hand, in IBC,

- knowing the `master-key`, the PKG can decrypt or sign the messages for any client, without any active attack and consequent detection,
- the PKG can make clients' private keys public without any possible detection, and
- in a validity period-based key revocation system [6], bringing down the PKG is sufficient to bring the system to a complete halt (single point of failure).

Therefore, with inherent key escrow and a single point of failure, the PKG in IBC needs to be far more trusted than the CA in a PKI, and this has been considered as a reason for the slow adoption of IBC schemes.

Boneh and Franklin [6] suggest the use of verifiable secret sharing (VSS) to solve this problem. They share the `master-key` among multiple PKGs using Shamir's secret sharing with a dealer [41] and also hint towards the use of the completely distributed schemes of Gennaro et al. [20]. In an *(n, t)-distributed PKG*, the `master-key` of the IBC system is distributed among n PKG nodes such that a set of nodes of size t or smaller has no information about the `master-key`, while a client extracts her private key by obtaining private-key shares from any $t + 1$ or more nodes; she can then use the system's public key to verify the correctness of her thus-extracted key. Although various proposed practical applications using BF-IBE, such as key distribution in ad-hoc networks [29] or pairing-based onion routing [28], require a distributed PKG to function correctly, there is no distributed PKG implementation available yet. This practical need for a distributed PKG that can function over the Internet forms the motivation of this paper.

Proactive Secret Sharing. It is well known that the most common attacks on security mechanisms are system attacks, where the system's cryptographic keys are directly exposed, rather than the cryptanalytic attacks more studied in the literature. Due to the endless supply of security flaws in almost all existing software, these system attacks are often easy to implement. In the previous paragraph, we saw that a distribution of trust among several system nodes, known as *threshold cryptography*, provides a solution to this problem. Although threshold cryptography enhances security against system break-ins, its effect is limited. Given sufficient time, a *mobile attacker* can break into system nodes one by one (*gradual break-in*) and eventually compromise the security of the whole system.

Proactive secret sharing [26], which combines distributed trust with periodic share refreshing, protects a system against these gradual break-ins. In a system with proactive secret sharing, the system's time is divided into phases. At the beginning of each phase, nodes' secret shares are refreshed such that new shares are independent of previous ones, except for the fact that they interpolate to the same system key. With an assumption from threshold cryptography that the adversary may corrupt at most t nodes in each phase, the system now becomes completely secure as shares from older phases are useless to the adversary.

Forward Secrecy. Threshold cryptography with proactive secret sharing assumes a weak correlation of the failures of the various system nodes. In practice, however, this is not likely. System nodes tend to employ a limited choice of software and operating systems. A real-world adversary that can break into t nodes using a software flaw can most likely break into one more node—or even all—exploiting the same flaw. Therefore, although threshold cryptography with proactive secret sharing is a robust and resilient solution in theory, we need a mechanism to reduce the losses to the minimum possible in the case of a complete system break-in.

In original proactive secret sharing, only the nodes' shares are refreshed at the end of each phase, while the system key remains the same. In such a setting, with a break-in into more than t nodes in a single phase, the adversary annihilates the security of the system retroactively since its inception. In order to reduce the severity of such an attack, proactive secret sharing scheme should have forward secrecy [16], such that even after a complete break-in, the adversary can only violate the security in that phase, but not any time prior to that. In this paper, we introduce the concept of forward secrecy in proactive secret sharing by amending the definition of the latter to include system key refreshing along with the refreshing of nodes' shares during a phase change.

Contributions. In this paper, we present the first practical architecture and implementation for a distributed PKG in the BF-IBE setup over the Internet. While doing it, we observe the importance of forward secrecy for a `master-key` in practical IBC-based applications, and provide a protocol for periodic `master-key` modification based on the secret share renewal protocol by Herzberg el al. [26]. We also enhance the adversary model in [26] and modify their share recovery protocol to be secure against a stronger adversary. Observing the importance of handling changes to groups in a realistic distributed PKG implementation, we devise protocols for group modification and security threshold modification. Finally, we demonstrate the efficiency of our system by comparing it with other VSS architectures proposed for the Internet and verify its practicality with a geographically distributed installation of our implementation.

Organization. Section 2 covers previous work related to PKG in IBC, proactive VSS and distributed key generation. In Section 3, we describe our assumptions and system model and based on this model a distributed PKG system is designed in Section 4. Forward secrecy and proactive security of the system are discussed in Section 5. In Section 6, we introduce the group modification protocols for our distributed PKG nodes. In Section 7, we summarize the important components of our implementation and compare its performance with alternative systems. We conclude and examine some interesting future work in Section 8.

2 Related Work

Key escrow and single point of failure are inherent to IBC. In their pioneer IBE scheme, Boneh and Franklin [6] suggested the use of threshold cryptography in the form of a distributed PKG to mitigate these problems. They however did not consider proactive security and forward secrecy of the `master-key`. Lee et al. [32] and Gangishetti et al. [19] propose variants of the distributed PKG involving a more trustworthy key generation centre (KGC) and other key privacy authorities (KPAs). But, as observed by Chunxiang et al. [14] for [32], these approaches are vulnerable to passive attack by the KGC, and thus insecure.

In parallel efforts, Gentry's certificate-based encryption [23] and Al-Riyami and Paterson's certificateless public key cryptography [2] address the key escrow problem by combining IBC with public-key cryptography (PKC) and consequently, sacrifice some of the important features of IBC. Recently, Goyal [25] reduced the required trust in the PKG by restricting its ability to distribute a client's private key. However, the PKG in his system still can decrypt the clients' messages passively, leaving a secure and practical implementation of a distributed PKG wanting.

The notion of secret sharing was introduced independently by Shamir [41] and Blakley [4] in 1979. Since then, it has remained an important topic in security research. Significantly, Chor et al. [13] introduced verifiability in secret sharing and Feldman [18] developed the first efficient and non-interactive protocol for it. Feldman proved the computational security for the secret and unconditional integrity of shares against a static adversary, which can only choose its t compromisable nodes before a protocol run. He also suggested a modification to the protocol to prove the security against adaptive adversary, but claimed [18, Sec. 9.3] that his original protocol is also secure against adaptive adversaries even though his simulation-based security proof did not work out.

Pedersen presented another VSS [38] with unconditional security for the secret but providing share integrity under a computational assumption. We observe that in most of the solutions based on Pedersen's VSS scheme (e.g. the distributed key generator (DKG) in [20]) the unconditional security for the secret is not achievable. Further, Pedersen's scheme also requires the random selection of generators g and h such that no player should know the relation between them; this adds an additional round of communication in practice. Therefore, with its simplicity

and efficiency, Feldman's VSS forms the basis for most of the VSS-based DKG systems in the literature and for the one in this paper, as well.

Pedersen also developed a completely distributed VSS-based DKG [37]. In this scheme, each node runs a variation on Feldman's VSS (with digital signatures and a commitment function) and distributed shares are added at the end to generate a combined shared secret. Gennaro et al. [20] observed that use of digital signatures and a commitment function does not provide any additional security to Pedersen's DKG and presented a simplification using just original Feldman's VSS called the Joint Feldman DKG (JF-DKG). They also claimed that these DKGs do not guarantee a uniformly random distribution of generated secret keys. In [21], the same set of authors proved the security of JF-DKG against a static adversary, when it is used with a provably secure Schnorr signature scheme. They also showed that JF-DKG produces hard instances of discrete logarithm problems (DLPs) and it can be used with other provably secure schemes [22, Sec. 5]. Although a reduction in their JF-DKG security proof is weak, as they discussed in [22], an elliptic curve implementation of JF-DKG with appropriately increased key sizes is still faster than the modification they suggested in [20].

As in Feldman's VSS, there is no known adaptive adversary attack against Pedersen's DKG, JF-DKG or the modification from [20]. They are not considered secure against an adaptive adversary only because their simulation-based security proofs do not go through when the adversary can corrupt players adaptively. [1, Sec. 3],[22, Sec. 4.4] Canetti et al. [10] presented a scheme provably secure against adaptive adversaries with at least two more communication rounds as compared to JF-DKG and with interactive zero-knowledge proofs. Recently, Abe and Fehr [1] developed an adaptively secure VSS without any interactive zero-knowledge proofs. However, an efficient bidirectional mapping [1, Sec. 4.2] required by their protocol seems to be difficult to obtain over the elliptic curves required in IBC. Further, as both of these adaptive (provably) secure protocols are far too inefficient for practical use, we use the efficient Feldman's VSS which has remained unattacked in the static as well as the adaptive adversary model for the last 20 years.

Herzberg et al. [26] proposed the first practical proactive secret sharing scheme, which can work with any VSS (including Feldman's VSS) based on homomorphic functions in the synchronous communication model. They provided schemes for share renewal and recovery and claimed their security against t compromised nodes in an (n, t)-DKG system with mobile adversary [36]. We modify their protocol to be secure against a stronger mobile adversary, and show that our modifications also thwart an attack on their system which was suggested by Nikov and Nikova [35]. Further, we also observe the practicality of the modified protocols in partially synchronous networks such as the Internet.

VSS schemes with unconditional security in the asynchronous communication model [9] also have been developed; however, they are prohibitively expensive for any realistic use. Recently, Cachin et al. [7] and Zhou et al. [47] suggested more practical proactive VSS schemes in the asynchronous communication model with some symmetric assumptions. Still, the proactive security protocol in APSS by Zhou et al., with $O(\binom{n}{t})$ message complexity, is not practical, while a share recovery protocol is missing in Asynchronous VSS by Cachin et al. Further, any practical completely distributed (dealerless) implementation of Asynchronous VSS by Cachin et al. requires at least $O(n^3)$ messages and five rounds of communication, which might be expensive for many systems. Note that two additional rounds over those suggested in [7] arise from the dealerless setup and distributed creation of group generators. Asynchronous VSS also does not generate the system's public key as required in IBC schemes; doing so may require another round of communication. Most importantly, forward secrecy for the shared secret, also of benefit to IBC, is not readily available and seems difficult to achieve. Indeed, forward secrecy for the shared secret is not considered by any other proactive VSS scheme in the literature.

As we intend our distributed PKG implementation to be useful over the Internet, a partially synchronized network, we use our modifications to the Herzberg et al. proactive VSS [26] rather than impractical [9, 47] or inapplicable [7] asynchronous VSS schemes.

3 Assumptions and System Model

In this section, we discuss the assumptions and the system model for our distributed PKG system, giving special attention to its practicality over the Internet.

Partially Synchronized System Model. Our distributed PKG system should be deployable over the Internet. The maximum message-transfer delay and the maximum clock offset there (a few seconds in general) is significantly smaller than the required timespan of our system's phase or the `master-key` modification (a few days). With such an enormous difference, a failure of the network to deliver a message within a small fixed time bound (in seconds) can be treated as a failure of the sender. This may lead to a retransmission of the message after appropriate timeout signals or immediate administrator intervention. As this is possible without any significant loss in the synchrony of the system, here we treat the Internet as a *partially synchronous* network [43].

Investigating further, we find that the Internet with a public-key infrastructure (PKI) and an online bulletin board [39] provides enough synchronization to securely implement a synchronous DKG protocol. Although messages can arrive in a partially synchronous manner, the `master-key` modification protocol as a whole proceeds in synchronized rounds of communication. Importantly, this avoids asynchronous VSS schemes with high computation and communication overheads. We note that Gennaro et al. [20, Sec. 5] make a similar observation for their modification to the JF-DKG protocol.

Bulletin Board with PKI. We use a PKI infrastructure to achieve authenticated and confidential communication, along with non-repudiability, and a bulletin board for message broadcasts. In our system, all messages on the bulletin board contain signatures by the entities posting them and all secure and authenticated communications between two nodes use the TLS protocol [15] with PKI certificates. The PKI infrastructure can be implemented as a PKI hierarchy or a web-of-trust; the online bulletin board can be any online service with atomic append and read operations on signed messages. Although we implemented our bulletin board from scratch, it could also be realized by using a broadcast mechanism such as SINTRA [8] or Secure Spread [3].

We require very limited trust in the certifying authority and the bulletin board. In particular, they cannot attack the system passively and any active attack—such as selective replies by the bulletin board—can be easily verified by the system nodes. Importantly, these central authorities do *not* learn the master secret or have a share of it. Further, in a PKI hierarchy with long-term certificates, a certifying authority does not have to be online except during system setup and rare group modifications. Similarly, even if a bulletin board becomes unavailable for a short period, the primary operation of distributed private-key generation remains unhampered.

Distributed Key Generation with Fault Tolerance. Although our system model can support any of the synchronous DKG schemes in the literature, we choose the simplest and most efficient: JF-DKG with $n \geq 2t + 1$, which requires at least two fewer rounds of communications than others. We only require $n \geq 2t + 1$ rather than the $n \geq 3t + 1$ needed for asynchronous systems robust against Byzantine failures [31] as the partially synchronous nature of the network, the timeout messages, the PKI infrastructure and the lack of a modification operation on the bulletin board provide sufficient synchronous behaviour, as mentioned earlier.

Our distributed PKG achieves proactiveness (with forward secrecy) through a periodic `master-key` modification protocol. In practice, this protocol is, though fast, not instantaneous and cannot be considered as an atomic action. We overcome this problem using a versioning system implemented with loosely synchronized local clocks to effect key refreshing and logical clocks [30] for the more tightly synchronized protocols.

Unlike in the asynchronous VSS by Cachin et al. [7], versions (or phases) in our system overlap with each other during executions of the `master-key` modification protocol. We expect these transition periods to be very small; they mainly involve PKG nodes and the bulletin board selecting a subset of modifications based on counting signatures and the nodes appending their signatures on the newly verified commitments on the bulletin board. Further, unlike [7], we do not assume that every node is honest at the start of each phase. As with [26], nodes may remain compromised in consecutive phases. We assume that the adversary is t*-limited* and can compromise up to t nodes; however, when one of those t compromised nodes is in the recovery phase (the share recovery and renewal protocol of Section 5.4), the adversary cannot capture another honest node until that node gets completely recovered. It may of course still demonstrate any malicious behaviour using up to $t - 1$ other compromised nodes. In the adversary model of [26], the adversary may only control at most t nodes over the complete timespan of a system phase. In our system, we assume a stronger and more practical adversary, which may control up to t nodes at any instant of time;

thus, over a system phase, our adversary may control more than t nodes. These modifications related to proactive security are further elaborated in Section 5.1.

Our protocol works for an IBC setting having a setup similar to that of BF-IBE, where, given the `master-key` s, the private key for an identity `ID` is $sH(\texttt{ID})$. (Here, H is a cryptographic hash function mapping identities to points in an elliptic curve group where DLP computations are hard.) A distributed computation of private keys for other IBC protocols involving more complicated private keys, such as $\frac{1}{s+H(ID)}U$, presents an interesting problem which we are currently investigating as future work.

Security Assumptions. We assume a polynomially bounded adversary which can wait for the messages of the uncorrupted players to be transmitted, then decide on his computation and communication for that round, and still get his messages delivered to the honest parties on time. As mentioned above, JF-DKG produces hard instances of the discrete logarithm problems. Thus, similar to JF-DKG with the Schnorr signature scheme, JF-DKG with a provably secure IBE scheme will most likely be provably secure against such an adversary; a complete proof remains as interesting future work.

Note that JF-DKG is provably secure against a static adversary. In a theoretical sense, it is not known to be secure against an adaptive adversary which may corrupt nodes adaptively and solutions requiring provable security against adaptive adversaries may use the DKG by Canetti et al. [10] without any significant modification to our model.

4 Distributed Private-key Generation

In this section, we present our core distributed PKG setup and the private key generation protocol. We utilize an improved Feldman VSS with a PKI infrastructure for PKG nodes and an online bulletin board to implement these protocols. We assume that a PKI infrastructure, with all nodes carrying their public-key certificates, and an online bulletin board application are already in place. We initiate the discussion by describing the cryptographic primitives and computational hardness assumptions we utilize in this paper.

4.1 Preliminaries

Bilinear Pairings. IBC extensively utilizes bilinear pairings over elliptic curves. For two additive cyclic groups $\mathbb{G}$ and $\hat{\mathbb{G}}$ and a multiplicative cyclic group $\mathbb{G}_T$, all of the same prime order q, a bilinear map e is a map $e : \mathbb{G} \times \hat{\mathbb{G}} \rightarrow \mathbb{G}_T$ with following properties.

1. **Bilinearity:** For all $P \in \mathbb{G}$, $Q \in \hat{\mathbb{G}}$ and $a, b \in \mathbb{Z}_q$, $e(aP, bQ) = e(P, Q)^{ab}$.
2. **Non-degeneracy:** The map does not send all pairs in $\mathbb{G} \times \hat{\mathbb{G}}$ to unity in $\mathbb{G}_T$.
3. **Computability:** There is an efficient algorithm to compute $e(P, Q)$ for any $P \in \mathbb{G}$ and $Q \in \hat{\mathbb{G}}$.

The IBC protocols under consideration in this paper, like many pairing-based cryptographic protocols, use a special form of pairing called a *symmetric pairing* which has $\mathbb{G} = \hat{\mathbb{G}}$. For such pairings, $e(P, Q) = e(Q, P)$ for any $P, Q \in \mathbb{G}$. The modified Weil pairing over elliptic curve groups [44] is an example of a symmetric bilinear pairing. In the rest of the paper, all bilinear pairings are symmetric.

Assumptions. In all the IBC schemes under consideration, $\mathbb{G}$ is a prime order subgroup of an elliptic curve over a finite field and $\mathbb{G}_T$ is a multiplicative subgroup of an extension of that finite field. The hardness of the computation Diffie-Hellman problem (CDH) over the groups $\mathbb{G}$ and $\mathbb{G}_T$ forms the main cryptographic assumption in this paper. It is a problem of computing abU given $U, aU, bU \in \mathbb{G}$ for $a, b \in \mathbb{Z}_q^*$. Specifically, the CDH assumption provides security for the `master-key` and nodes' shares given the public parameters of the system. However, we observe that proving the security of the IBC protocols themselves, using keys generated by our PKG, assumes the hardness of bilinear Diffie-Hellman problem (BDH) [6]. This problem involves the computation of $e(U, U)^{abc} \in \mathbb{G}_T$, given

$U, aU, bU, cU \in \mathbb{G}$ for $a, b, c \in \mathbb{Z}_q^*$. Further, in verification of private keys, a client utilizes the fact that if the pairing computation is feasible for the group $\mathbb{G}$, the Decision Diffie-Hellman problem (DDH) over the group $\mathbb{G}$ can be solved in polynomial time. For $U \in \mathbb{G}$, the problem of distinguishing $\langle U, aU, bU, abU \rangle$ from $\langle U, aU, bU, cU \rangle$ for $a, b, c \in_R \mathbb{Z}_q^*$ is known as the DDH problem. Joux and Nguyen [27] observe that the DDH problem over $\mathbb{G}$ is easy, as $c = ab \bmod q \Leftrightarrow e(U, cU) = e(aU, bU)$.

BF-IBE Setup with Distributed PKG. In a BF-IBE setup [6], a trusted authority called a private key generator (PKG) generates private keys (d) for clients using their well-known identities (`ID`) and a `master-key` s. A client with identity `ID` receives the private key $d_{\texttt{ID}} = sH(\texttt{ID}) \in \mathbb{G}$, where $H : \{0,1\}^* \rightarrow \mathbb{G}^*$ is a full-domain cryptographic hash function. In such a setup, with its knowledge of s, the PKG has the key escrow property and is a single point of failure for the BF-IBE scheme. Boneh and Franklin propose the use of the DKG by Gennaro et al. [20] to distribute their PKG. Here, we improve this distributed PKG setup and present a practical protocol for the same.

As already suggested, JF-DKG provides a simpler and more efficient distributed key generation solution than the new DKG by Gennaro et al. [20]. In JF-DKG, all n nodes initiate the VSS and the qualified ones contribute to the `master-key` generation. With the t threshold assumption, there is at least one honest node in any group of $t + 1$ nodes. Therefore, in a `master-key` generation step, we just need $t + 1$ qualified VSSs. Further, as we elaborate in Section 5, to achieve proactive security and forward secrecy, we similarly never need more than $t + 1$ qualified VSSs. In such a scenario, given a large timeframe of a system phase as compared to the average time to complete a VSS, a node can asynchronously and at any time during the phase decide whether or not to make a contribution to the `master-key`. It can make this decision based on the current system state, such as the number of currently qualified modifications, or its trust of nodes that have already contributed. Although in many practical systems each node could take part in every `master-key` modification, this selective modification can be used for added flexibility. We thus extract an improved Feldman VSS from the JF-DKG protocol for use in our system, rather than using the latter protocol in its entirety.

4.2 Improved Feldman VSS over an elliptic curve group

We next describe an improved version of the Feldman VSS over an elliptic curve group, which provides the basis for all of our protocols discussed henceforth. Modifications to the original Feldman VSS in this scheme, which are quite similar to those in JF-DKG, include converting the dealer into an ordinary node and the use of signatures on all parameters published on a bulletin board with only append and read operations allowed. These modifications to the Feldman VSS add only to the practicality of its implementation over the Internet and do not hamper the security of the protocol while achieving this.

We seek an (n, t)-distributed key generation setup over an elliptic curve group $\mathbb{G}$ of order q and generator U. Let $F(z) = a_0 + a_1 z + \cdots + a_t z^t \in \mathbb{Z}_q[z]$ be the current shared polynomial and $s = a_0$ be the currently shared secret. Note that no player—not even the bulletin board application—knows $F(z)$ or s. Let $s_j = F(j)$ be the secret share possessed by node P_j for $j = 1, \ldots, n$, and $\texttt{Pub} = sU$ be the corresponding system public key. We create an online bulletin board containing verified and signed (by the system nodes) commitments $A_k = a_k U$ for $k = 0, \ldots, t$. Note that A_0 is the system public key. Figure 1 presents our improved Feldman VSS over the elliptic curve group $\mathbb{G}$, when P_i is a node interested in updating the shared secret s.

4.3 Distributed PKG Setup

Now, using our improved Feldman VSS we present a protocol for the distributed PKG setup. It involves distributed creation of the system's `master-key` with generation of secret shares for n nodes.

1. Given a security parameter κ, the bulletin board application chooses a prime q of size κ, two groups $\mathbb{G}$ and $\mathbb{G}_T$ of order q, and a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. The bulletin board application chooses a cryptographic hash function $H : \{0,1\}^* \rightarrow \mathbb{G}^*$ and a few other scheme-specific parameters. It also chooses a random generator $U \in \mathbb{G}$ and sets up the bulletin board by publishing the above system parameters and by initializing

Improved Feldman VSS

1. Node P_i chooses a random polynomial $f_i(z) \in \mathbb{Z}_q[z]$ of degree t:

$$f_i(z) = a_{i0} + a_{i1}z + \cdots + a_{it}z^t$$

and broadcasts (on the bulletin board) signed copies of $A_{ik} = a_{ik}U$ for $k = 0, \ldots, t$. Here, $x_i = a_{i0} = f_i(0)$ and $Y_i = a_{i0}U = f_i(0)U$ are P_i's new contributions to the `master-key` s and the corresponding public key sU respectively. It then computes a subshare $s_{ij} = f_i(j) \bmod q$ for $j = 1, \ldots, n$ and sends it to node P_j over a secure and authenticated TLS connection.

2. Node P_j verifies the subshare s_{ij} received from the node P_i by checking whether

$$s_{ij}U \stackrel{?}{=} \sum_{k=0}^{t} j^k A_{ik} \tag{1}$$

If the verification succeeds, P_j publishes a confirmation signature for P_i's A_{ik} values. If it fails, P_j broadcasts an accusation against P_i. If more than t nodes accuse P_i, the suggested modification to s is disqualified and the protocol stops. If not, for each accusing party P_j, node P_i broadcasts the corresponding s_{ij}, such that (1) holds. If any of the revealed subshares fails, the suggested modification to s by P_i is disqualified and the protocol for P_i stops. Otherwise, P_j keeps the new s_{ij}, erasing the old one.

3. If there is no disqualification, after $t + 1$ success messages, including one from P_i itself, the modification $f_i(z)$ suggested by P_i is ready to be acquired by the system. However, the actual acquisition time depends upon the protocol using this improved Feldman VSS.

4. At the time of acquisition, the bulletin board application computes new A_k values for $k = 0, \ldots, t$ as $A_k + A_{ik}$ and appends them to the bulletin board with a new phase number. Nodes acquire and verify these commitments, and after successful verifications, publish signatures on those values. After observing $t + 1$ or more signatures on the new A_k commitments, each node P_j for $j = 1, \ldots, n$ modifies its secret share $s_j = s_j + s_{ij}$. As a result, the shared secret and the corresponding public key becomes $s = s + x_i$ and $\texttt{Pub} = \texttt{Pub} + Y_i = A_0$ respectively.

5. Anybody can compute the public-key share for the node P_j as $\texttt{Pub}_j = \sum_{k=0}^{t} j^k A_k$.

6. To achieve forward secrecy and proactiveness, once the new secret share s_j is computed, node P_j erases s_{ij} and node P_i erases $f_i(z)$ and s_{ij} for $j = 1, \ldots, n$.

Figure 1: Improved Feldman VSS over an elliptic curve group

the A_k and A_{ik} values to zero for $i = 1, \ldots, n$ and $k = 0, \ldots, t$. Consequently, the `master-key` s is set to zero.

2. Nodes interested in contributing to s initiate the improved Feldman VSS defined in Figure 1. As an adversary can compromise a maximum of t nodes, once $t + 1$ or more nodes successfully finish their protocol runs, the distributed shares are considered safe. In the literature, such nodes are known as the *qualified* nodes. We denote their set as $\mathcal{Q}$.

3. The bulletin board then computes and broadcasts the coefficients A_k (for $k = 0 \ldots t$) for the implied shared polynomial $F(z) \cdot U$ as $A_k = \sum_{P_i \in \mathcal{Q}} A_{ik}$.

4. After verifying the new A_k values, nodes send confirmation signatures to the bulletin board.

5. On receiving $t + 1$ or more confirmation signatures, the A_k values are finalized. Each node then computes their secret share as $s_i = \sum_{P_j \in \mathcal{Q}} s_{ji}$. They also obtain a signed (by $t + 1$ or more nodes) copy of A_k values from the bulletin board.

4.4 Private-key Extraction

After a successful setup, PKG nodes are ready to extract private keys for system clients. We assume that at least $t+1$ out of the n nodes will be available to the client. Let $\mathcal{O}$ be a set of $t+1$ online servers chosen by a client. The private-key extraction protocol works as follows.

1. A client with identity $\texttt{ID}$ contacts online nodes from the set $\mathcal{O}$.
2. Each node $P_i \in \mathcal{O}$ verifies the client's identity and returns a private-key share $s_i H(\texttt{ID})$ over a secure and authenticated channel.
3. Upon receiving $t+1$ correct shares of her private key, the client can construct her private key $d_{\texttt{ID}}$ as $d_{\texttt{ID}} = \sum_{P_i \in \mathcal{O}} \lambda_i s_i H(\texttt{ID})$, where the Lagrange coefficient $\lambda_i = \prod_{P_j \in \mathcal{O} \setminus \{i\}} \frac{j}{j-i}$.
4. The client can verify the correctness of the computed private key $d_{\texttt{ID}}$ using the polynomial-time solvability of the DDH problem over the elliptic curve group. In particular, she can check if $e(d_{\texttt{ID}}, U) \stackrel{?}{=} e(H(\texttt{ID}), \texttt{Pub})$. If unsuccessful, she can verify the correctness of each received $s_i H(\texttt{ID})$ by checking if $e(s_i H(\texttt{ID}), U) \stackrel{?}{=} e(H(\texttt{ID}), \texttt{Pub}_i)$. An equality proves the correctness of the share, while an inequality indicates misbehaviour by the node P_i and a consequential complaint.

4.5 Security Analysis

This system setup is quite similar to that of JF-DKG, although it is far less synchronized. Both of the protocols present the same structure just before `master-key` generation and we consider our system as secure as JF-DKG. Gennaro et al. [22, Sec. 5] provides a detailed security analysis of JF-DKG and prove that JF-DKG produces hard instances of the discrete-log problem. They also prove the security of the JF-DKG-based threshold Schnorr signature scheme and suggest that security of other DLP-based schemes can be proved similarly. We observe that the security of our distributed private-key generation can be proved in a similar way.

5 Realizing Proactiveness

In threshold cryptography, part of a system might come under an attacker's control. Further, a mobile attacker can gradually break into other nodes in the system in order to gain control over the complete system. As we discuss in the introduction, periodic refreshing of the shares as well as the secret itself is therefore important in maintaining both the system's existing security as well as forward secrecy. To realize proactive security in a distributed key generation environment, we need protocols for secret share renewal and recovery, and for periodic secret modification. In this section, we investigate the existing proactive security solutions and provide efficient practical implementations for the same. In particular, we investigate and modify the adversary model for the share renewal and recovery protocols by Herzberg et al. [26] and present secure protocols for periodic `master-key` modification and secret share recovery in the stronger model.

5.1 Mobile Adversary Model

Herzberg et al. [26] introduce the first proactive VSS scheme. They provide periodic secret share renewal and recovery schemes for any DKG based on the Feldman or the Pedersen VSS and show that their protocols for (n,t)-distributed key generation systems are secure against a mobile adversary controlling up to t nodes in a system phase.

In our DKG, to achieve forward secrecy, the `master-key` is modified along with the nodes' shares of it at each phase change. A stronger adversary model, with the adversary able to control up to t nodes at any given instant, rather than t nodes over the complete timespan of a system phase, is more appropriate in a practical setting where phase durations are in days. To obtain security against this stronger adversary, nodes' shares are also renewed in the middle of a phase, when a compromised node requires recovery (see Section 5.4). Thus, a recovery protocol in our model involves a combination of a share recovery for the recovering node and a share renewal (initiated by the

recovering node) for the whole group. An adversary compromising t nodes is eligible to compromise an additional node only after the recovery protocol for one of those t nodes completes.

5.2 Attacks on the Proactive VSS by Herzberg et al.

In an adversary model similar to one defined above, Nikov and Nikova [35] proposed an attack on the share renewal scheme of Herzberg et al. They observe that the share renewal scheme used at the phase change is only secure against an adversary controlling up to $t-1$ nodes. In that protocol, the polynomials of degree t that are used have a fixed constant term of 0; an adversary controlling only t nodes (rather than the claimed $t+1$) can reconstruct these polynomials and nullify any sought proactiveness.

This attack does not work on our protocol, as the modification polynomials at a phase change have random constant terms. The attack is also not applicable during a share renewal of a recovering node, as at that time, the adversary can control only up to $t-1$ nodes. We observe that in the share recovery scheme of Herzberg et al., the temporary modification polynomial (say $h(z) \in \mathbb{Z}_q[z]$) has a special property that $h(r) = 0$, where r is the index for the receiving node. However, there too, if the adversary controls t nodes including the recovering one, it cannot compute the shares for honest nodes even though it can compute h.

5.3 Periodic `Master-key` Modification

In this protocol, each node interested in contributing to the `master-key` modification for the next system phase asynchronously starts secret modification using the improved Feldman VSS protocol from Figure 1. At the end of the current phase (v), the modifications from all the successful protocol runs are selected and added to the current commitments ($A_k^{(v)}$ values for $k = 0, \ldots, t$) to obtain the commitments ($A_k^{(v+1)}$ values) for the phase $(v+1)$.

Unlike other share renewal schemes, at a phase change, we modify the `master-key` along with shares. This provides the desired forward secrecy for the `master-key` which is an important requirement in many practical applications in identity-based settings. A modification to the `master-key` by a compromised node does not provide any forward secrecy, while that by any honest node preserves forward secrecy for the whole system. The adversary can compromise up to t nodes and it might not be possible for an honest node to determine which nodes are compromised. Therefore, to achieve forward secrecy, at the end of our periodic `master-key` modification protocol, we expect modifications by at least $t+1$ nodes to be incorporated into the `master-key`.

Although performing a new DKG protocol from scratch in each phase provides equivalent forward secrecy, a cumulative mechanism of adding modifications (instead of replacing the secret completely) certainly has an added advantage. In our adversary model, were we to use a separate DKG protocol per system phase, we would need $t+1$ qualified VSS invocations for even proactive security. However, by using a cumulative approach to modify the `master-key`, proactive security can be achieved by t or fewer qualified VSSs. Here, even if an adversary knows all of the qualified VSSs, she cannot determine the `master-key`. When she leaves one node to compromise another, the share recovery protocol also renews the shares for the whole system.

We next present the actual modification protocol. We assume that the system is in phase (v) and enters in the next phase $(v+1)$ after a predefined period (usually measured in days).

1. Nodes interested in modifying the `master-key` for the next version $(v+1)$ initiate the improved Feldman VSS protocol (Figure 1) sometime during the current phase.

2. At the time of transition, the bulletin board application picks the successful protocol runs. Let $\mathcal{Q}$ be the set of such qualified nodes. It then computes $A_k^{(v+1)} = A_k^{(v)} + \sum_{P_i \in \mathcal{Q}} A_{ik}$ for $k = 0, \ldots, t$ and sends these values to all the nodes.

3. Nodes verify and sign the set $\mathcal{Q}$ and the $A_k^{(v+1)}$ values. Once these elements are approved by $t+1$ nodes, each node P_i computes its share for phase $(v+1)$ as $s_i^{(v+1)} = s_i^{(v)} + \sum_{P_j \in \mathcal{Q}} s_{ji}$.

Security Analysis. As already discussed, an adversary in our protocol does not know any root or coefficient of the modification polynomials $f_i(z) \in \mathbb{Z}_q[z]$ chosen by honest nodes P_i; thus, it cannot compute the polynomial $f_i(z)$ with only up to t shares known to him. Therefore, if at least one honest node is involved in every run of the periodic `master-key` modification protocol, it is forward secure against such an adversary. This can be easily formally proved by closely following the security proof of the share renewal protocol by Herzberg et al. [26, Sec. 3.3]

Further, we avoid the attack in [35] on the share renewal protocol by Herzberg et al., as our periodic `master-key` modification involves secret modification along with share renewal.

5.4 Secret Share Recovery

In a proactive distributed PKG implementation, the ability of a node to recover its lost share must be ensured. Otherwise, the adversary can destroy the complete system by gradually corrupting $n - t$ nodes. Once a corruption is detected, the corresponding node has to be rebooted in a trusted way, using read-only memory for instance. As a next step, other nodes in the system need to assist the rebooted node to recover its secret share and here, we present the share recovery scheme to accomplish this in our adversary model.

This secret share reconstruction is based on Herzberg et al.'s share recovery and share renewal schemes. In our model, we assume that the attacker can compromise more than t nodes during a system phase, though it is restricted to t nodes at any instant. Thus, the adversary may be controlling $t - 1$ other nodes, and may know the secret share for a t^{th} node, which is currently under recovery. After a successful recovery of the t^{th} node, it can compromise another node. In order to avoid the attacker thus learning $t + 1$ shares (although controlling only t compromised nodes), we combine a secret share renewal protocol along with the secret share recovery.

Assume a rebooted node P_r wants to restore its current secret share $s_r^{(v)}$ and any $t + 1$ or more helper nodes with correct secret shares choose to help it. Let $\mathcal{H}$ represent a set of these nodes. In the secret share recovery scheme, nodes in $\mathcal{H}$ choose polynomials $h_i(z) \in \mathbb{Z}_q[z]$ of degree t such that $h_i(r) = 0$, run the improved Feldman VSS protocol (Figure 1) and provide their new temporary shares to P_r. P_r can then compute its secret share $s_r^{(v)}$ by interpolating $t + 1$ shares, without knowing anything about the helper nodes' original shares. To achieve share renewal without `master-key` modification, P_r chooses a polynomial $g_r(z) \in \mathbb{Z}_q[z]$ of degree t such that $g_r(0) = 0$ and runs the improved Feldman VSS protocol to update the system polynomial $F^{(v)}(z) = F^{(v)}(z) + g_r(z)$. This achieves immediate proactive security without changing the master secret mid-phase.

In detail:

1. Each node $P_i \in \mathcal{H}$ initiates the improved Feldman VSS protocol (Figure 1) with a degree t polynomial $h_i(z) \in \mathbb{Z}_q[z]$ such that $h_i(r) = 0$:
$$h_i(z) = b_{i0} + b_{i1}z + \cdots + b_{it}z^t.$$

2. Node P_r starts the improved Feldman VSS protocol (Figure 1) with a degree t polynomial $g_r(z) \in \mathbb{Z}_q[z]$
$$g_r(z) = c_{r0} + c_{r1}z + \cdots + c_{rt}z^t.$$
such that $c_{r0} = 0$.

3. In the protocol runs started by $P_i \in \mathcal{H}$, at least t other nodes should verify that $h_i(r) = 0$ by checking if $\sum_k r^k(b_{ik}U) = 0$. (Note that the $(b_{ik}U)$ were published as part of the improved Feldman VSS.) Similarly, in the protocol run started by P_r, at least t nodes should verify that $g_r(0) = 0$ by checking if $(c_{r0}U) = 0$.

4. Once protocol runs by any $t + 1$ nodes in $\mathcal{H}$ succeed, each of the successful nodes P_i computes its temporary share $r_i = s_i^{(v)} + \sum_{j\in\mathcal{H}} h_j(i)$ and sends it over a secure channel to P_r.

5. P_r verifies the received r_i values using the public key shares $Pub_i^{(v)}$ and broadcasted $b_{ik}U$ as follows.
$$r_iU \stackrel{?}{=} \left(s_i^{(v)} + \sum_{j\in\mathcal{H}} h_j(i)\right)U = Pub_i^{(v)} + \sum_{j\in\mathcal{H}}\left(\sum_{k=0}^{t} i^k(b_{jk}U)\right)$$

After verification, P_r interpolates the correct shares to compute its share $s_r^{(v)}$.

6. Once the protocol run initiated by P_r succeeds, the bulletin board application computes $A_k^{(v)} = A_k^{(v)} + \sum_{P_i \in \mathcal{Q}} C_{ik}$ for $k = 0, \ldots, t$. After verifying the new $A_k^{(v)}$ values, all the nodes can compute their new secret shares $s_i^{(v)} = s_i^{(v)} + s_{ri}$

7. Nodes in $\mathcal{H}$ should compute their new shares only after the temporary shares they sent are verified by P_r. This local synchronization does not affect the asynchronous nature of the overall protocol. It is possible to make this share renewal independent or asynchronous to the temporary share verification by P_r. However, that requires the nodes in $\mathcal{H}$ inform P_r whether a temporary share sent by them contains the renewed value or not. In this description, we avoid this intricacy. Further, the `master-key` and the system public key are not modified by this protocol and consequently, clients' private keys do not need any modifications.

Security Analysis. In our above protocol, as we use only the share renewal and recovery schemes in [26], its security can be seen as a straightforward extension to the security of those share renewal and recovery schemes.

6 Group Modification Protocols

In a practical distributed PKG system, on a long term basis, it is inevitable that the set of nodes in the PKG group will need to be modified; new nodes may join the system or old nodes may leave. A modification to the total number of nodes may also lead to a modification in the security threshold value of the system. In this section, we present protocols to achieve group modification and security threshold modification in our distributed PKG. These protocols are based on the share renewal and recovery schemes by Herzberg et al. and their security can be proved in a similar way.

6.1 Group Modification

In a group modification, we are changing the set of PKG nodes without modifying the threshold parameter (t). This alters the redundancy parameter (n) of the distributed PKG, changing the number of nodes that can be unavailable without affecting the operation of the PKG. On the other hand, this does not change the security parameter (t), which controls how many nodes an attacker can compromise without learning any information about the `master-key`. Next, we define protocols to add as well as to remove a node from the group.

Node Addition. As expected, the process to add a new node to the system is similar to the process to retrieve a share for a recovering node that had lost it. However, in this case, we can safely assume that the adversary does not a priori know the share for the node to be added. This removes the requirement of running a share renewal protocol for the node being added. As with node recovery, we assume that the adversary cannot capture the node which is getting added until the share generation for that node completes. The rest of the share recovery protocol is followed as-is.

Node Removal. This protocol involves removing a node from the group, such that it should neither be able contribute to the `master-key` nor possess a share of it. The best way to remove a node from the group is to simply not provide it with subshares for the next phase's `master-key` modification. By doing this, the node may continue to perform private-key extractions for clients during the current phase, but as new modifications become active in the next phase, it gets eliminated from the group. It is also possible to remove a node without waiting for the end of a phase. We achieve this through an execution of the share renewal protocol. In this protocol, $t + 1$ nodes from the group, other than the one getting eliminated, start share renewal protocols and provide subshares to all nodes except the one getting removed. Once these share renewals qualify, all other nodes add the new subshares to their shares and the node getting removed becomes ineffective.

6.2 Security Threshold Modification

Security threshold modification involves changing the threshold limit t for the system, without changing the total number of nodes n. This amounts to changing the degree of the system polynomial, which requires changing the polynomial itself.

Security Threshold Addition. Given an (n, t)-distributed PKG, we may wish to convert it into a $(n, t + \Delta)$-distributed system; this increases the number of nodes that may be compromised without revealing the `master-key`. We achieve this easily by increasing the degree of the next phase's `master-key` modification polynomials by Δ. Once any of those modifications qualifies at the end of the phase, the system's threshold limit will be increased to $t + \Delta$. It is also possible to perform this modification mid-phase using the share renewal protocol; nodes would use polynomials of degree $t + \Delta$ with constant term 0. Here, however, we need to insist that the adversary remain t-limited during the execution of this protocol based on share renewal. Once the the renewals by $t + 1$ or more node succeeds, system becomes secure against an adversary controlling $t + \Delta$ nodes.

Security Threshold Reduction. An elegant way for reducing the security parameter t for the system presents an interesting challenge. We achieve this by running the node addition protocol for an index γ of an nonexistent node to publicly obtain the share $F^{(v)}(\gamma)$. We then note that $F^{(v)}(z) - F^{(v)}(\gamma)$ is a polynomial of degree t, one of whose roots is γ, and so $(z-\gamma)$ is a factor. Dividing by $(z-\gamma)$ yields the desired system polynomial $\hat{F}(z)$ of degree $t-1$. Nodes perform following steps to achieve the security threshold reduction.

1. Using the node addition protocol for an index γ of an nonexistent node, publicly compute its share $F^{(v)}(\gamma)$.
2. Each node modifies its individual share to $\hat{s_i} = (s_i - F^{(v)}(\gamma))(i-\gamma)^{-1}$. Since the old shares satisfied $s_i = F^{(v)}(i)$, the new shares satisfy $\hat{s_i} = \hat{F}(i)$, and $\hat{F}(z)$ is of degree $t-1$.

Note that this protocol assumes that the adversary is already $t-1$-limited during the protocol run and it also modifies the `master-key` during the process. It is therefore only suitable to be used at the end of a phase. It is of course also possible to perform a security threshold reduction by throwing away all previous information about shares of the distributed PKG and starting over with polynomials of reduced degree.

7 System Architecture and Implementation

We design our distributed PKG as a deterministic state machine. Our object-oriented C++ implementation uses the PBC library [33] for the underlying elliptic curve and finite field operations and a PKI infrastructure with DSA signatures based on GnuTLS [34] for confidentiality, authentication and non-repudiation. In this section, we briefly discuss the design and implementation of our distributed PKG.

7.1 System Design

State Machine. In our distributed PKG state machine, we have a bulletin board, n PKG nodes, and numerous clients as system entities. The bulletin board can be in a non-functional or functional state, while a node can be in a non-functional, under-recovery or functional state. The non-functional states for the bulletin board and nodes indicate their behaviour before initialization, while the functional state indicates their possession of a certificate, a corresponding private key and an appropriate identifying index: zero for the bulletin board and greater than zero for the nodes. A node may also be in the under-recovery state, which indicates that it has been rebooted after a compromise, but does not yet possess its `master-key` share. Nodes in the under-recovery and functional states are included in the *active nodes* list and participate in `master-key` modification; however, nodes in the under-recovery state cannot be helpers to assist in the recovery of other nodes (see Section 5.4).

class CommitmentVector{	**class CommitmentEntry**{	**class Signature**{
SystemParam sysParam; CommitmentType cType; NodeID vectorID; unsigned int phase; map <NodeID, CommitmentEntry> entries; ⋮ };	NodeID committerID; EntryTag tag; unsigned short numcommits; element_t *commitments; map <NodeID, Signature> signatures; *//Node's private Subshare* element_t subshare; unsigned short confirmationCnt; unsigned short accusationCnt; ⋮ };	NodeID signerID; SignatureType signType; unsigned int timestamp; string DSA; bool accusedSubsharePresent; element_t accusedSubshare; string accusedDSA; ⋮ };

Figure 2: Commitment Data Structure.

PKI. We assume that a PKI in the form of a PKI hierarchy or a web-of-trust is already present for the bulletin board and all the PKG nodes. The entities use this PKI to perform authenticated and secure communication over TLS [15] links. In addition to using these PKI certificates for client and server authentication in TLS, some individual messages are also DSA-signed, in order that they may be of use in proving another party's malfeasance.

Messages. Messages in our state machine can be categorized into three types: user messages, network messages and timer messages. A user message, involving a node or a bulletin board application and its user, assists the user in monitoring the application as well as in checking a correct behaviour by other entities. Network messages realize all protocol flows in the system among the bulletin board, node, and client applications. As the Internet is not synchronized (even with TLS links, which can go down, resulting in dropped messages at the application layer), we also include timer messages. For query-response types of network messages, requesters use pre-defined timeout messages to resend their queries, as we consider any network or adversary failure as a sender failure. We also use timer messages to implement end of phase messages, where after a pre-defined interval the bulletin board and nodes perform end of phase tasks involving the modification of the `master-key` and its shares. The time interval for timeouts as well as for the end of phase messages are installation parameters.

Data Structures. The most important data structure in our application is the **CommitmentVector**, the most important pieces of which are indicated in Figure 2. This data structure encapsulates commitment entries—the $a_{ik}U$ values from step 1 of Figure 1—and signatures confirming the consistency of those commitments from a number of other nodes.

7.2 Dataflow for Distributed Key Generation

Figure 3 depicts the message flow for the core distributed key generation process. It involves the messages related to `master-key` modification, end of phase, and private-key extraction. Here, subshare proposals are sent by their generating nodes to the appropriate peer nodes. The peers verify whether the subshares they received are correct (see step 2 of Figure 1). If they are, the peers send a *confirmation signature* to the bulletin board; otherwise, they send an *accusation signature*. Similarly, nodes use signed messages to they approve or disapprove of the final `master-key` modifications chosen by the bulletin board. The Accused Subshare Request, Accused Subshare and Verified Subshare messages are optional and can only occur when there are accusations.

Although not pictured here for brevity, the actual application also supports additional message flows for system initiation, node recovery, and group and security threshold modification.

7.3 System Issues

Bulletin Board. We currently implement our bulletin board using the above-defined CommitmentVector data structure and DSA signatures over TLS links. It is possible to use other broadcast messaging systems such as

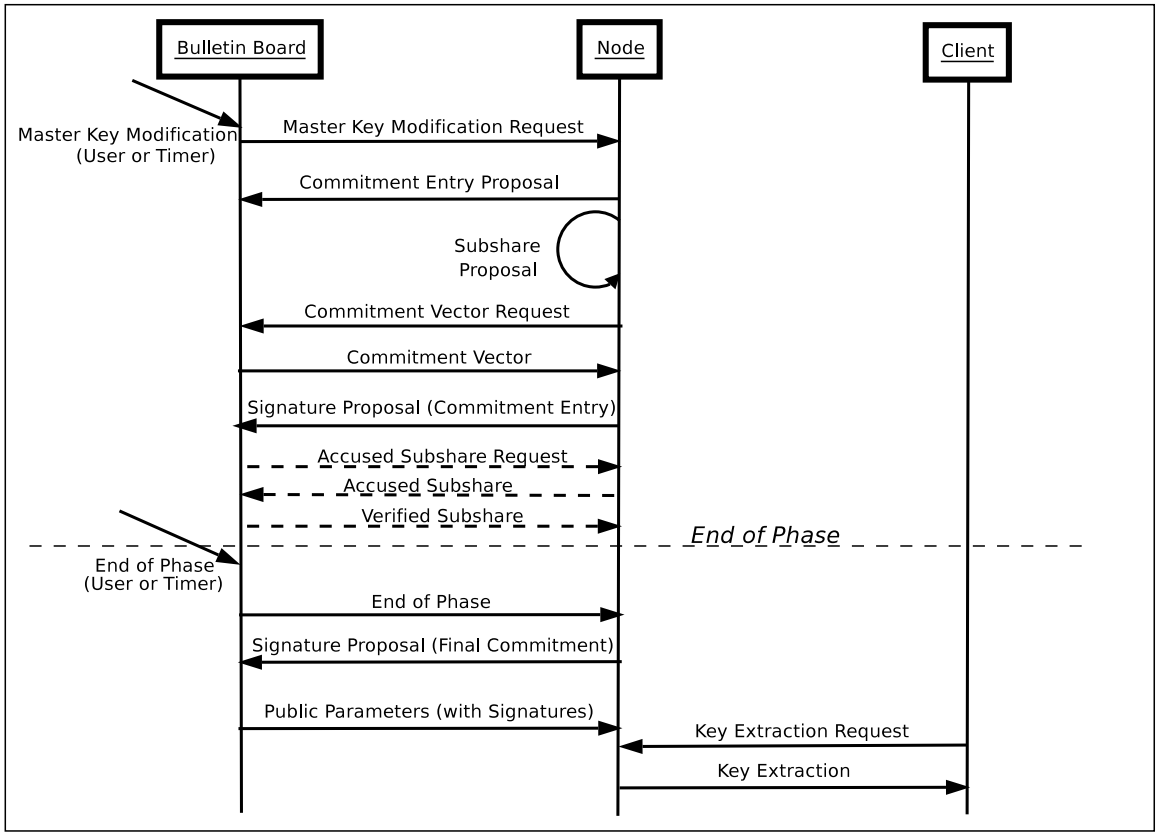

Figure 3: Distributed Key Generation Message Flow.

SINTRA [8] or Secure Spread [3] in place of our bulletin board. However, as only read and append operations need to be supported, a simpler system like the one in our current application, which also supports a mechanism for accusations against the malfunctioning of the bulletin board itself, suffices for the purpose.

DoS attacks and Sybil attacks. The (n, t)-distributed nature of the PKG nodes gives an inherent protection against DoS attacks on the nodes. Bringing down the bulletin board is more problematic, but is not a fatal problem, as the primary distributed private-key extraction mechanism continues to work properly, even if the bulletin board is not available for a short period. Further, this problem also can be mitigated using a more decentralized broadcast systems.

Neither are Sybil attacks [17] a major concern, as the ad-hoc addition of a node is not a feature of this system. In practice, additions of new nodes will require confirmation by the PKI, if not a number of the existing nodes.

7.4 Performance Analysis

Efficiency and simplicity have been important design criteria in our architecture and subsequently, the message and communication complexity for our protocol is the lowest among the proactive VSS designs for the Internet. We start our discussion on performance analysis by comparing the theoretical complexities of our implementation with other proposed distributed PKG designs.

Table 1 compares our distributed PKG with other distributed PKG designs: the proactive secret sharing of Herzberg et al. [26], asynchronous VSS by Cachin et al. [7] and APSS by Zhou et al. [47]. We observe that the message complexity of our distributed setup and proactive security protocols is lower than that of asynchronous VSS and APSS. Although our message complexity is equivalent to that of [26], we obtain advantage by introducing forward secrecy with a periodic `master-key` modification protocol. Indeed, none of the proactive secret sharing

Operation	**This paper**	Herzberg et al. [26]	Cachin et al. [7]	Zhou et al. [47]
Distributed PKG setup	$O(n^2)$	$O(n^2)$	$O(n^3)$	$O(n\binom{n}{t})$
Forward secrecy	$\surd$	X	X	X
Availability of the public key	$\surd$	$\surd$	X	X
Periodic `master-key` modification	$O(n^2)$	-	-	-
Secret share renewal	$O(n^2)$	$O(n^2)$	$O(n^3)$	$O(\binom{n}{t})$
Secret share recovery	$O(n^2)$	$O(n^2)$	-	$O(\binom{n}{t})$
Group Modification	$O(n^2)$	-	-	-
Threshold Modification	$O(n^2)$	-	-	-

Table 1: Comparison among distributed PKG designs

schemes before this considered forward secrecy. Further, we improve on the proactive VSS by Herzberg et al. [26] by introducing group and security threshold modification protocols and by avoiding unnecessary signatures.

To test the efficiency of our implementation, we installed a bulletin board, five PKG nodes and a number of clients on Linux and FreeBSD machines at various locations in Canada, the United States and Europe. We created a $(5, 2)$-distributed PKG system and measured the computational performance of the important operations, such as commitments and share generation by nodes, share verification and signature generation by the receiving nodes, and end of phase processing by the bulletin board. For 80-bit security, all these operations take just few milliseconds of CPU time to complete. The `master-key` modification protocol requires just half a second of wall-clock time to receive and process contributions from all five nodes, while the bulletin board's end of phase processing is just 10 ms of wall-clock time. Finally, our client application takes less than one second of wall-clock time to request key extraction from each of the nodes, receive their replies, and reconstruct and verify her private key. These measures demonstrate that our system is eminently practical. Further, the large difference between the duration of a system phase and that of an execution of our implementation justifies our assumptions about the partially synchronized nature of the system and the design of our protocols.

8 Conclusion

We have presented the first practical implementation for a distributed PKG for identity-based cryptography. Our architecture uses a verifiable bulletin board and a PKI to implement the distributed PKG in a partially synchronized network such as the Internet. In our simple proactive design with a stronger adversary model, we have provided efficient protocols for non-traditional group and threshold modification operations, along with traditional share secret renewal and recovery protocols. Further, we observed the importance of maintaining forward secrecy of the shared `master-key` and have provided the first protocol in the secret sharing literature that combined periodic `master-key` modification with intra-phase proactive share updating. We also compared the message complexity of our protocol with other proactive distributed PKG designs and demonstrated the efficiency and practicality of our distributed PKG for real-world IBC-based applications. Finally, we installed our application in a geographically distributed environment, and verified its efficiency and the practicality of our system assumptions.

Going forward, we would like to produce a more generic distributed PKG. Our distributed PKG does not handle some of the recent IBC schemes having setups different from that of BF-IBE. Considering the potential applications of these new IBC schemes, we find the problem of providing them with a distributed PKG to be exciting future work.

References

[1] M. Abe and S. Fehr. Adaptively Secure Feldman VSS and Applications to Universally-Composable Threshold Cryptography. In *Advances in Cryptology—CRYPTO'04*, pages 317–334, 2004.

[2] S. S. Al-Riyami and K. G. Paterson. Certificateless Public Key Cryptography. In *Advances in Cryptology—ASIACRYPT'03*, pages 452–473, 2003.

[3] Y. Amir, C. Nita-Rotaru, J. R. Stanton, and G. Tsudik. Secure Spread: An Integrated Architecture for Secure Group Communication. *IEEE Transactions on Dependable and Secure Computing*, 2(3):248–261, 2005.

[4] G. R. Blakley. Safeguarding cryptographic keys. In *the National Computer Conference*, pages 313–317, 1979.

[5] D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology—EUROCRYPT'05*, pages 440–456, 2005.

[6] D. Boneh and M. K. Franklin. Identity-Based Encryption from the Weil Pairing. In *Advances in Cryptology—CRYPTO'01*, pages 213–229, 2001.

[7] C. Cachin, K. Kursawe, A.Lysyanskaya, and R. Strobl. Asynchronous verifiable secret sharing and proactive cryptosystems. In *ACM Conference on Computer and Communications Security*, pages 88–97, 2002.

[8] C. Cachin and J. A. Poritz. Secure intrusion-tolerant replication on the internet. In *DSN*, pages 167–176, 2002.

[9] R. Canetti. *Studies in Secure Multiparty Computation and Applications*. PhD thesis, The Weizmann Institute of Science, 1996.

[10] R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Adaptive Security for Threshold Cryptosystems. In *Advances in Cryptology—CRYPTO'99*, pages 98–115, 1999.

[11] J. Cha and J. Cheon. An identity-based signature from gap Diffie-Hellman groups. In *Public Key Cryptography*, pages 18–30, 2003.

[12] Liqun Chen and Caroline Kudla. Identity based authenticated key agreement protocols from pairings. Technical report, 2002. `http://eprint.iacr.org/2002/184`.

[13] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults (Extended Abstract). In *26th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 383–395, 1985.

[14] X. Chunxiang, Z. Junhui, and Q. Zhiguang. A Note on Secure Key Issuing in ID-based Cryptography. Technical report, 2005. `http://eprint.iacr.org/2005/180`.

[15] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol (Version 1.1), Request for Comments (RFC) 4346. `http://www.ietf.org/rfc/rfc4346.txt`.

[16] W. Diffie, P.C. van Oorschot, and M. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2:107–125, 1992.

[17] J. R. Douceur. The Sybil Attack. In *International Workshop on Peer-to-Peer Systems (IPTPS '02)*, pages 251–260, 2002.

[18] P. Feldman. A Practical Scheme for Non-interactive Verifiable Secret Sharing. In *28th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 427–437, 1987.

[19] R. Gangishetti, M. Choudary Gorantla, M. Das, and A. Saxena. Threshold key issuing in identity-based cryptosystems. *Computer Standards & Interfaces*, 29(2):260–264, 2007.

[20] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Advances in Cryptology—EUROCRYPT'99*, pages 295–310, 1999.

[21] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure Applications of Pedersen's Distributed Key Generation Protocol. In *CT-RSA*, pages 373–390, 2003.

[22] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. *Journal of Cryptology*, 20(1):51–83, 2007.

[23] C. Gentry. Certificate-Based Encryption and the Certificate Revocation Problem. In *Advances in Cryptology—EUROCRYPT'03*, pages 272–293, 2003.

[24] C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In *Advances in Cryptology—ASIACRYPT'02*, pages 548–566, 2002.

[25] V. Goyal. Reducing Trust in the PKG in Identity Based Cryptosystems. In *Advances in Cryptology—CRYPTO'07*, pages 430–447, 2007.

[26] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive Secret Sharing Or: How to Cope With Perpetual Leakage. In *Advances in Cryptology—CRYPTO'95*, pages 339–352, 1995.

[27] A. Joux and K. Nguyen. Separating Decision Diffie-Hellman from Computational Diffie-Hellman in Cryptographic Groups. *Journal of Cryptology*, 16(4):239–247, 2003.

[28] A. Kate, G. M. Zaverucha, and I. Goldberg. Pairing-Based Onion Routing. In *7th Privacy Enhancing Technologies Symposium (PET)*, pages 95–112, 2007.

[29] A. Khalili, J. Katz, and W. Arbaugh. Toward Secure Key Distribution in Truly Ad-Hoc Networks. In *IEEE Workshop on Security and Assurance in Ad-Hoc Networks 2003*, pages 342–346, 2003.

[30] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.

[31] L. Lamport, R. E. Shostak, and M. C. Pease. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[32] B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang, and S. Yoo. Secure key issuing in ID-based cryptography. In *ACSW Frontiers '04: Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation*, pages 69–74, 2004.

[33] B. Lynn. PBC Library – The Pairing-Based Cryptography Library. `http://crypto.stanford.edu/pbc/`, 2008. Accessed January 2008.

[34] N. Mavroyanopoulos, F. Fiorina, T. Schulz, A. McDonald, and S. Josefsson. The GNU Transport Layer Security Library. `http://www.gnu.org/software/gnutls/`, 2008. Accessed January 2008.

[35] V. Nikov and S. Nikova. On Proactive Secret Sharing Schemes. In *Selected Areas in Cryptography*, pages 308–325, 2004.

[36] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks (extended abstract). In *10th Annual ACM Symposium on Princiles of Distributed Computing (PODC)*, pages 51–59, 1991.

[37] T. P. Pedersen. A Threshold Cryptosystem without a Trusted Party. In *Advances in Cryptology—Eurocrypt'91*, pages 522–526. Springer-Verlag, 1991.

[38] T. P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Advances in Cryptology—CRYPTO'91*, pages 129–140, 1991.

[39] R. A. Peters. A Secure Bulletin Board. Master's thesis, Technische Universiteit Eindhoven, 2005.

[40] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Symposium on Cryptography and Information Security (SCIS 2000)*, 2000.

[41] A. Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979.

[42] A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *Advances in Cryptology—CRYPTO*, pages 47–53, 1984.

[43] J. Shamsi, C. Chu, and M. Brockmeyer. Towards partially synchronous overlays: Issues and challenges. In *AAA-IDEA*, pages 10–17, 2005.

[44] E. Verheul. Evidence that XTR Is More Secure than Supersingular Elliptic Curve Cryptosystems. In *Advances in Cryptology—Eurocrypt'01*, pages 195–210, 2001.

[45] B. Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology—EUROCRYPT'05*, pages 114–127, 2005.

[46] T. Yuen and V. Wei. Constant-size hierarchical identity-based signature/signcryption without random oracles. Technical report, 2005. `http://eprint.iacr.org/2005/412`.

[47] L. Zhou, F. B. Schneider, and R. van Renesse. APSS: proactive secret sharing in asynchronous systems. *ACM Trans. Inf. Syst. Secur.(TISSec)*, 8(3):259–286, 2005.