Blind Bernoulli Trials: A Noninteractive Protocol For Hidden-Weight Coin Flips

Emma Connor and Max Schuchard, *University of Tennessee*

https://www.usenix.org/conference/usenixsecurity19/presentation/connor

This paper is included in the Proceedings of the
28th USENIX Security Symposium.

August 14–16, 2019 • Santa Clara, CA, USA

978-1-939133-06-9

Blind Bernoulli Trials: A Noninteractive Protocol For Hidden-Weight Coin Flips

Emma Connor and Max Schuchard

University of Tennessee

Abstract

We introduce the concept of a "Blind Bernoulli Trial," a noninteractive protocol that allows a set of remote, disconnected users to individually compute one random bit each with probability *p* defined by the sender, such that no receiver learns any more information about *p* than strictly necessary. We motivate the problem by discussing several possible applications in secure distributed systems. We then formally define the problem in terms of correctness and security definitions and explore possible solutions using existing cryptographic primitives. We prove the security of an efficient solution in the standard model. Finally, we implement the solution and give performance results that show it is practical with current hardware.

1 Introduction

Distributed systems sometimes require users to make random choices to drive network behavior. For example, peer-to-peer anonymous systems such as Freenet [12], AP3 [26], and DiscountANODR [36] employs a random "coin flip" in routing decisions to help obscure information about the path of a request from an observer. Opportunistic routing protocols may use a random decision on whether to forward or cache certain content or not. Systems that do rely on users making coin flips usually model the coin flip as a random trial that produces a single bit with a fixed probability. They distribute the probability of the trial's two possible outcomes either as a static, pre-defined parameter known to the whole network, or as a dynamic parameter distributed to users as cleartext.

However, in some instances the trial probability itself may be sensitive information. For example, if we use dynamic trial probabilities to prioritize certain content in a network (i.e., have some content forwarded with higher probability than others), then observers can distinguish and target higher-priority content. If an anonymous communication system varied the probability of forwarding to fine-tune the trade-off between performance and anonymity for individual messages, then a malicious node could selectively attempt to deanonymize easier traffic.

For such applications we can envision a cryptographic solution that allows each user to carry out only a single trial and obtain a random bit with some weighted probability, while learning as little as possible about the overall probability of each outcome. The user should not be able to repeat the trial for a different result, since users can easily approximate the probability using multiple results. Nor should they be able to use the trial parameters to learn anything more about the actual probability of the outcomes. Users should be able to perform a trial noninteractively, as they would if the probability were distributed in cleartext. In other words, we want a way to distribute a weighted coin that each user can flip once, while revealing *as little information as possible* about the weight of the coin. We call this construction a "Blind Bernoulli Trial, or *BBT*."

Specifically, we propose a definition where an authority generates and distributes unique keys to individual users. For each trial, the authority generates an encrypted *tag* that corresponds to the desired probability for that trial. Given a user key and a tag, one can noninteractively compute the outcome of exactly one trial without learning the overall probability.

We formalize the security of this system with a simulation-based definition inspired by the usual definition of semantic security for a cipher. Informally, the definition states that any function that can efficiently be computed by some number of identities and trial parameters could also be computed only knowing the trial results. The definition also includes a "leakage function" to allow schemes that leak some information but still have near-ideal security. The leakage function formally quantifies and places an upper bound on the amount of information an attacker can gain.

This paper evaluates three BBT schemes. First we develop a very simple protocol that meets our definitions and is based on a semantically secure cipher. This scheme essentially encrypts one trial result per user. We discuss why this trivial solution is unsatisfactory and then show an alternative construction from a general functional encryption primitive. Finally, since no practical functional encryption scheme for general functions is known, we examine more specific functional encryption schemes that support only a limited class of functions. We show how to construct a near-ideal BBT scheme from functional encryption supporting only inner product predicate functions, for which practical schemes currently exist.

In section 6, we compare the security of the ideal BBT schemes with the inner product construction using a quantitative attack analysis, discussing what an attacker can learn about the trial probability given a certain number of trial results. We design and evaluate a simulation of an attacker's perspective on both possible schemes, using the different information available to the adversary in each case. The attack simulations show that the information gained by an attacker for the inner product scheme, on average, is very similar to the information gained in the ideal case.

Since efficiency is a major concern, we discuss both the running time and storage requirements for the inner product scheme. To evaluate the feasibility of current inner product functional encryption schemes, we implement a recently proposed scheme in software (to our knowledge, the first implementation of this scheme) and provide benchmarks for each algorithm involved in a Blind Bernoulli trial scheme. The benchmarks show that a Blind Bernoulli Trial scheme based on inner product encryption can run in a reasonable amount of time on current hardware.

Finally, we explore in more detail some potential applications of this new cryptographic concept. We discuss two possible distributed-systems scenarios where random behavior is used and the probability of that random behavior is sensitive information. In these cases, Blind Bernoulli trials can enhance privacy in the distributed system by hiding the weighted probabilities from users.

1.1 Related Work

Protocols for remote parties to agree on a random bit in a way that is fair and verifiable go back decades in cryptography [6]. These protocols differ from BBT in that their goal is to prevent either party from biasing the result. BBT is almost the opposite: here we explicitly want one party to be able to bias the result and for the other party to be unable to determine the bias.

More generally, secure multi-party computation (MPC) encompasses a wide body of related work that deals with allowing remote parties to interactively perform arbitrary computations together. MPC focuses on protocols that enable distrusting parties to jointly compute a function on private inputs, without revealing the inputs to each other. It allows for private inputs from both parties, and protocols are interactive, proceeding in multiple rounds. Existing MPC schemes can be practical [31]. Our formulation of BBT does not allow interactive protocols, and the only private input is the probability of the trial. Therefore BBT is not compatible with MPC solutions.

While BBT does not fall under the area of MPC, it does fit squarely within the functional encryption model. Section 4 gives additional background on functional encryption and shows how BBT can be instantiated using general functional encryption.

In contrast with MPC, no practical general functional encryption is known. Recent works have proposed general functional encryption schemes, although these are not yet practical [15]. Other works have focused on implementing efficient functional encryption for specific classes of functions such as inner products and polynomials [18]. This paper primarily focuses on building an efficient construction specifically for BBTs.

2 Blind Bernoulli Trials

A Bernoulli trial models a random process with two possible results, where each result occurs with a fixed probability. This has applications in some distributed systems. For example, it provides a very simple means for one user to direct the behavior of a certain percentage of others without knowing exactly how many there are and without needing direct communication. An authority can distribute the parameters for a trial, and users can run a Bernoulli trial with the given parameters to self-organize into groups of approximately the desired proportions.

However, in some cases it may be important to the security goals of the system that individual users do not learn the overall probability of success. In these cases it is not acceptable for an authority to distribute the parameters for a trial, since this directly reveals the overall probability of success to all users. In response to this need, we formulate the concept of a *Blind Bernoulli Trial*, or *BBT*, which allows each user to obtain a single pseudo-random trial result without revealing additional information about the overall probability of success.

At first glance, it might appear that trivial solution would be for the authority to run the trials on a trusted computer and individually send a different trial result to each user. Since a user sees only their result, this scheme is secure. However, this scheme does not meet the requirement that a BBT be noninteractive. This leaves it with an important drawback compared to an unencrypted Bernoulli trial (publishing the probability parameter in plaintext). For an unencrypted Bernoulli trial, the authority can publish the probability parameter once, and any number of users can run a trial or forward the trial parameters to other users. Instead, the trivial interactive solution forces the authority to open an individual communication channel for each user. As a result, this interactive solution presents scalability concerns for systems where communication between the authority and users may be intermittent or costly.

Ideally, a BBT scheme should more closely mirror the properties we get with a noninteractive unencrypted Bernoulli trial. The authority should be able to publish an encrypted object that represents the trial parameters and any number of users should be able to use this object to obtain a trial result without further interaction with the authority. Therefore, a Blind Bernoulli Trial scheme will try to construct "tags" that represent encrypted trial parameters of varying probabilities. Users will be able to use these tags to conduct trials without interaction with the authority.

In order to hide the overall probability of success for a trial, users must be able to obtain only one trial result per tag. If users could run multiple pseudo-random trials with the same tag they could quickly approximate the probability of success. To avoid this, we require that trial results are deterministic on a per user basis. In other words, the same user will always compute the same result for a given tag. Since Blind Bernoulli Trials must be deterministic, they are not true Bernoulli trials and do not have a "probability" of success in the same sense. Instead, when using a BBT, we are more interested in the overall probability of a trial's success across a distribution of users. Accordingly, when we speak of the "probability" of an outcome of a Blind Bernoulli Trial, we are referring to the probability of that outcome when a trial is performed with a user key that is selected at random from the set of users. For schemes that require the authority to store key material for each user, the "set of users" refers to the set of user keys kept by the authority. Otherwise, it refers to the set of all possible user keys.

Just as a single user must always get the same result for the same tag, it is also necessary to prevent one entity from controlling many user identities and utilizing them to perform multiple trials on a tag. For this reason, it must be impractical for an adversary to create multiple working user identities. We avoid this by introducing a master key that is required to generate new user identities. These identities take the form of a user key, which is combined with the tag to conduct a single trial. In general, an adversary should not be able to create a user key without the master key. The impact of collusion is discussed at length in Section 6.

Taking into account these properties, we can arrive at a clearer picture for what our scheme must look like: in a cryptographic Blind Bernoulli Trial scheme, an *authority* uses a private *master key* to generate and distribute *user keys* representing individual user identities and *tags*, each representing a Bernoulli trial with a fixed probability of success prescribed by the authority. Given a user key and a tag, there exists a public, deterministic procedure to compute the result of a single Bernoulli trial (either "success" or "failure") without revealing to the user the probability of success associated with the trial.

Formally, a Blind Bernoulli Trial encryption scheme consists of the following algorithms:

- $\mathsf{Setup}(1^{\lambda})$: Accepts a security parameter λ and returns a master key sk and public parameters pk.
- $\mathsf{KeyGen}(\mathsf{sk})$: returns a user key uk.
- $\mathsf{TagGen}(x)$: takes a probability parameter x and returns a tag t; the exact form of the probability parameter can vary depending on the construction. In order to be useful, there must be at least two possible probability parameters that create tags with different probabilities of success.
- $\mathsf{Trial}(\mathsf{uk},\mathsf{t})$: returns a single bit b indicating success or failure.

This definition does not allow the trivial interactive solution mentioned earlier, where the authority carries out trials and directly communicates results to each user individually. This reflects a key design goal: that users must be able to obtain trial results without online communication with a centralized infrastructure. Users must be able to transfer tags to each other and each tag must be usable by all users. This allows individuals in a disconnected distributed system to obtain trial results without needing a direct intermediary.

2.1 Security Definition

Inherently, a BBT must reveal some information about the underlying priority. For example, an adversary that seeks to distinguish high-probability trials from low-probability ones could, after generating a trial result for a tag with their user key, guess "high-probability" for successful trials and "low-probability" for unsuccessful ones. Such a trivial adversary could already achieve non-negligible advantage in distinguishing between two types of tags.

Since each trial result unavoidably reveals *some* information about the underlying probability of success (a single successful trial means that the trial is more likely to have a higher probability of success), our security definition must take into account this inherent information leakage. Also, our definition must take into account collusion, so that the scheme remains as secure as possible even when a single adversary controls multiple user identities. Therefore, we compare the information an adversary learns from some number of user keys to that learned by an adversary that learns only the trial results corresponding to those keys.

Informally, a Blind Bernoulli Trial scheme is secure if an adversary with access to x user keys and y tags learns no more about the probabilities of success of any tags than he would by being given only the results of x trials for each tag. Since a BBT scheme intends to reveal the outcome of 1 trial per key, clearly this is the best any scheme could hope to do. In Section 6 we discuss some possible attacks when an adversary controls multiple keys and quantify the amount of information gained by such an adversary.

Formally, we use a simulation-based definition to capture the idea that any function which is efficiently computable from a trial tag and a set of user keys must also be efficiently computable using only the trial results. We also include some allowance for additional information leaked, as this will be useful later in constructing a scheme that achieves near-ideal security with much greater efficiency compared to other schemes.

Definition 2.1 (Security with leakage)**.** *A Blind Bernoulli trial scheme is secure with respect to a leakage function $\mathscr{L}$ if for all probabilistic polynomial time (PPT) algorithms $\mathscr{A}$, there exists a PPT algorithm $\mathscr{B}$ such that for all polynomially-bounded functions f,h, the advantage of $\mathscr{A}$, defined as:*

$$\begin{aligned}&\Pr[\mathscr{A}(1^\lambda,\mathsf{uk}_1,\mathsf{uk}_2,\ldots,\mathsf{uk}_n,\mathsf{t},h(1^\lambda,x)))=f(1^\lambda,x)]-\\&\Pr[\mathscr{B}(1^\lambda,\mathsf{uk}_1,\mathsf{uk}_2,\ldots,\mathsf{uk}_n,\mathsf{Trial}(\mathsf{uk}_1,\mathsf{t}),\mathsf{Trial}(\mathsf{uk}_2,\mathsf{t}),\ldots,\\&\qquad\mathsf{Trial}(\mathsf{uk}_n,\mathsf{t}),\mathscr{L}(\mathsf{t}),h(1^\lambda,x))=f(1^\lambda,x)]\end{aligned}\tag{1}$$

is negligible in the security parameter, where x is the probability parameter and $\mathsf{t} = \mathsf{TagGen}(x)$.

This definition is closely-related to the usual definition of semantic security for private-key encryption and formalizes the idea that an adversary should learn as little as possible about a tag beyond the results of the trials of all keys known to the adversary. The leakage function places an upper bound on the amount of information that an adversary can learn from a tag because the definition states that any function that can be efficiently computed with the tag t can also be efficiently computed with only the trial results (which are intentionally revealed) and $\mathscr{L}(\mathsf{t})$.

Implicit in this security definition is the design requirement that an adversary can not forge additional user keys. If a BBT system allowed an adversary to forge a non-zero number of additional keys, that adversary would gain access to an extra set of trial results beyond those generated from their originally controled keys. Such a system fails to meet our security definition.

2.2 Other Design Goals

Security is a necessary property, but it is not the only design goal. To be usable, a BBT scheme must be efficient, both in terms of the running times of the algorithms and the space complexity of keys and tags. As stated previously, we also require that the protocol is noninteractive; that is, that tags can be freely transmitted from user to user and that users can obtain trial results from a tag without direct communication with the authority.

Each algorithm must be efficient enough to run in a reasonable amount of time. The running time of the Trial algorithm is particularly important, as we expect this algorithm to be run most frequently. Each user must run a trial for each tag received. Also, several applications of BBT feature users with lower computing resources compared to the authority. The other algorithms that comprise a BBT scheme are likely to be run less often: Setup is run only once, or only when the system needs to be re-keyed. And if n tags are created and m users then we expect the number of trials run to be on the order of nm if most users receive most tags.

The size of objects in the scheme must also be efficient. “Efficient” tags and user keys should require space logarithmic or at least sublinear in the number of users. In order to minimize storage requirements for the authority, we would also prefer that the tag generation algorithm does not depend on the current state of users. This eliminates the need for the authority to keep a database of users, and also allows user keys to be used even with tags that were generated before the key. This is particularly important in distributed systems applications that are disconnected or high churn, where new users may regularly encounter tags that were generated before the user key.

Another potentially desirable property would be the ability for users to generate tags. We consider this property desirable because if it is not wanted it can easily be removed by composing tags with any cryptographic signature scheme. Users can then simply reject tags that do not have a valid signature from the authority. On the other hand, it is not clear how to add this property to a scheme that does not support it, so we consider a scheme that does allow user tag generation to be more flexible.

A less-obvious but important property of a BBT scheme is the possible probability values for a tag. There is no requirement that a scheme support an arbitrary probability, but only that TagGen accepts some parameter that increases or decreases the probability of success for trials resulting from the generated tag. A scheme that allows more fine-grained control of the probability level is preferable over one that supports more limited probability levels.

3 Construction from Semantically-Secure Encryption

A simple BBT scheme can be trivially constructed from any symmetric or asymmetric encryption scheme that is semantically secure. In short, the authority can simply generate and store a random key for each user and send a tag consisting of a different ciphertext for each user, which that user can decrypt to obtain a trial result with the corresponding user key. To run a trial, users simply decrypt the ciphertext corresponding to their key. The security of this scheme follows immediately from the semantic security of the underlying encryption system.

Either a public-key or symmetric system can be used here, as it long it meets the definition for semantic security. A symmetric-key system will be especially efficient, but a public-key system has the advantage that users can generate tags themselves. On the other hand, if a symmetric-key system is used, then the same keys that create tags can also decrypt them, which means that only the authority can hold the keys needed to create tags.

The individual algorithms are described as follows:

Setup

The authority initializes sk as an empty list of encryption keys.

Generating User Keys

The authority generates a decryption key uk for the underlying cryptosystem, gives it to the user, and appends its corresponding encryption key to sk (in the case of a symmetric system, the encryption key may be the same as the decryption key).

Generating Tags

A single tag consists of a set of ciphertexts, with one ciphertext per user. The authority generates it as follows:

1. The authority randomly selects a subset S containing x of $|\mathsf{sk}|$ users.

2. For each uk_i in sk, the authority computes $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}_{\mathsf{uk}_i}(m_{\mathsf{success}})$ if $\mathsf{uk}_i \in S$, otherwise $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}_{\mathsf{uk}_i}(m_{\mathsf{fail}})$

3. The tag is a tuple of all ct_i: $\mathsf{t} \leftarrow (\mathsf{ct}_1, \mathsf{ct}_2, \ldots, \mathsf{ct}_{|\mathsf{sk}|})$

4. The probability of success for the tag is $x/|\mathsf{sk}|$.

Trials

To perform a trial, a user selects the ciphertext corresponding to that user's key from the set of ciphertexts that forms the tag. The user then decrypts that ciphertext to obtain the trial result:

1. $m \leftarrow \mathsf{Decrypt}_{\mathsf{uk}_i}(\mathsf{ct}_i)$.

2. Return 1 if $m = m_{\mathsf{success}}$.

3. Otherwise, return 0.

3.1 Discussion

Since a trial consists only of a single decryption, trials are very efficient. User key generation is likewise extremely efficient as it requires only choosing a random key. Trials and user key generation are both $O(1)$. However, generating a tag is linear in the number of users, requiring l encryptions for l users. The space complexity of tags is also $O(l)$. When the number of users is known to be small, this may be acceptable. However, especially because BBT schemes are designed for applications where network resources are extremely limited, the linear space complexity of tags may quickly become a concern as the number of users increases.

This type of BBT scheme also allows for the most fine-grained possible control of probability. The tag generator can select any subset of users of any size for a successful trial. This is contrast to the schemes proposed in Sections 4 and 5, which are both limited in the possible subsets of users that observe a successful trial.

This scheme does not meet the design goal that tag generation does not depend on user state. The authority must maintain a central database of all user keys. If an asymmetric key system is used to allow tag generation by users, this key storage burden is also placed on each user. Each tag will be valid only for the user keys that existed in the authority's database at the time the tag was generated, so it will not be possible for newly-created users to run older tags.

4 Construction From Functional Encryption

A BBT scheme with ideal security can be constructed from any functional encryption scheme that supports arbitrary functions. In functional encryption, given a key k and ciphertext ct, one can learn the output of a function of the plaintext $f_k(m)$ without learning anything else about the plaintext [8, 30, 2]. To construct a BBT scheme from a functional encryption primitive, we define the user key functions using a pseudorandom function family (PRF) and a comparison. The tag plaintext consists of a random seed and a threshold value which determines the tag's likelihood of success. The authority encrypts tags under the functional encryption scheme and distributes the ciphertexts to users. Each user key corresponds to a function f that is defined as:

$$\begin{aligned} f(s,t) &= 1 \text{ if } h(s) < t \\ &= 0 \text{ otherwise} \end{aligned} \tag{2}$$

where h is a function selected at random from a PRF for each user key. Although the domain and range of functions in PRFs are typically viewed as bit strings, for our purposes it is more convenient to view them as integers in binary representation.

Setup

The authority initializes the functional encryption scheme and retains the master key sk which allows the creation of function keys.

Generating User Keys

The authority selects h at random from a PRF and generates the user key uk as the function key for f, as described above.

Generating Tags

The authority selects a seed s at random from the domain of each function in the PRF. The threshold t controls the probability p of the tag and is computed as $p * \mathsf{max}(\mathsf{range}(h))$. The authority then encrypts the tuple (s,t) under the functional encryption scheme to obtain the ct.

Trials

To perform a trial, a user computes $f(s,t)$ using the tag ct and the user key uk. The trial result is the function output.

4.1 Discussion

Although this construction achieves best-case security and allows fine-grained choice of success probabilities, its description relies on functional encryption for arbitrary functions. While such schemes do exist, they in turn rely on other heavy-handed approaches such as fully homomorphic encryption for which practical implementations are not yet available [15]. Therefore, a more practical solution is needed.

5 Construction from Inner Product Encryption

In this section we show how to use any fully attribute-hiding inner product encryption (IPE) scheme to construct a BBT scheme that is secure with respect to the leakage function $\mathscr{L}(\mathsf{t}) = \mathsf{pk}$, where pk is the public key of the IPE scheme used.

5.1 Background

The term "inner product encryption" has been applied to multiple related but distinct cryptographic concepts [21, 5, 28, 29, 27, 19, 3], In this context, we use it to refer to a specific form of predicate encryption where ciphertexts and keys are each associated with vectors, and the associated predicate is the inner product function. Predicate encryption is a generalized form of public key encryption where each key k is associated with a predicate function f_k, and each ciphertext is associated with an attribute y [18]. A ciphertext with attribute y can be decrypted with key k if and only if $f_k(y)$ is true.

In general, an IPE scheme operates on n-dimensional vectors of integers modulo a prime p. In an IPE scheme, each key $\mathsf{sk}_{\vec{k}}$ is associated with a vector $\vec{k} \in \mathbb{Z}_p^n$, and each ciphertext $\mathsf{ct}_{\vec{y}}$ is associated with an attribute vector $\vec{y} \in \mathbb{Z}_p^n$. The associated predicate is $f_{\vec{k}}(\mathsf{ct}_{\vec{y}}) = \vec{k} \cdot \vec{y} \overset{?}{=} 0$. In other words, given $\mathsf{sk}_{\vec{k}}$ and a ciphertext $\mathsf{ct}_{\vec{y}}$, one can compute the plaintext m if and only if $\vec{k} \cdot \vec{y} = 0$. An IPE scheme consists of the following functions:

- $\mathsf{Setup}(1^\lambda)$ outputs public key pk and secret key sk.
- $\mathsf{KeyGen}(\vec{k}, \mathsf{sk})$ accepts the secret key sk and a vector $\vec{k} \in \mathbb{Z}_p^n$ and outputs $\mathsf{sk}_{\vec{k}}$.
- $\mathsf{Encrypt}(m, \vec{y}, \mathsf{pk})$ outputs $\mathsf{ct}_{\vec{y}}$.
- $\mathsf{Decrypt}(\mathsf{ct}_{\vec{y}}, \mathsf{sk}_{\vec{k}}, \mathsf{pk})$ outputs m if $\vec{k} \cdot \vec{y} = 0$, otherwise outputs $\perp$.

Attribute-hiding IPE additionally requires that the vector $\vec{y}$ associated with each ciphertext is hidden. Partially attribute hiding schemes hide $\vec{y}$ from users who are not authorized to decrypt the associated ciphertext, while fully attribute-hiding schemes hide $\vec{y}$ even in the case where $\vec{k} \cdot \vec{y} = 0$.

IPE security is defined by a game between a challenger and an adversary [28].

Definition 5.1 (Attribute-hiding IPE Security). *The security of a fully attribute-hiding IPE scheme is defined by the following game between the challenger and an admissible adversary $\mathscr{A}$*

1. *The challenger runs* $\mathsf{Setup}_{\mathsf{IPE}}$ *and gives* pk *to* $\mathscr{A}$*, retaining* sk.
2. $\mathscr{A}$ *adaptively makes any polynomial number of key queries for key vectors* $\vec{k}_i$*. The challenger gives* $\mathscr{A}$ $\mathsf{sk}_{\vec{k}_i} \leftarrow \mathsf{KeyGen}(\vec{k}_i, \mathsf{sk})$
3. $\mathscr{A}$ *chooses challenge attribute vectors* $(\vec{y}_0, \vec{y}_1)$ *and challenge plaintexts* (m_0, m_1),.
4. *The challenger randomly selects a bit* $b = 0$ *or* $b = 1$.
5. *The challenger gives* $\mathscr{A}$ $\mathsf{Encrypt}(m_b, \vec{y}_b, \mathsf{pk})$
6. $\mathscr{A}$ *can again adaptively make a polynomial of key queries for additional key vectors* $\vec{k}_i$.
7. $\mathscr{A}$ *outputs a guess* b' *and wins the game if* $b' = b$.

Here, an admissible adversary is defined as one whose queries adhere to at least one of the following conditions:

1. $\vec{k}_i \cdot \vec{y}_0 \neq 0$ and $\vec{k}_i \cdot \vec{y}_1 \neq 0$ for all $\vec{k}_i$

2. $m_0 = m_1$ and either $(\vec{k}_i \cdot \vec{y}_1 \neq 0$ and $\vec{k}_i \cdot \vec{y}_0 \neq 0)$ or $(\vec{k}_i \cdot \vec{y}_0 = 0$ and $\vec{k}_i \cdot \vec{y}_1 = 0)$ for all $\vec{k}_i$.

Without these restrictions an adversary can trivially infer b by submitting a challenge attribute pair that will be possible to decrypt for one value of b and not possible to decrypt for another, or a challenge message pair that can be decrypted to a different value depending on b.

5.2 Construction

With attribute-hiding IPE and its security now defined, we can show how to construct a BBT scheme using it. Intuitively, we will construct user keys and tags from randomly sampled vectors. Tags will correspond to IPE ciphertexts, user keys correspond to IPE user keys, and trials correspond to IPE decryptions. A successful decryption means a successful trial, while a failed decryption indicates a failed trial. We will vary the number of nonzero components in a tag's associated vector to control the probability that a randomly-selected user key will be able to decrypt it. Because the IPE scheme is fully attribute-hiding, the vector associated with tags is hidden from users, regardless of trial result.

Setup

The Setup function for IPE-based BBT additionally accepts a paramter a that determines the number of nonzero components in each user key. The authority runs the following procedure:

1. Run $\mathsf{Setup}_{\mathsf{IPE}}(1^\lambda, n)$ to obtain the private key sk and public key pk.

2. Store a as a public parameter.

3. Select m_{success} randomly from the message space of the underlying IPE scheme, and store it as a public parameter.

Generating User Keys

Every user key has the same number of nonzero components, which is parameterized as a.

1. $\vec{k}$ is randomly selected from the set of all vectors with a nonzero entries.

2. uk is computed as $\mathsf{KeyGen}_{\mathsf{IPE}}(\vec{k}, \mathsf{sk})$

Generating Tags

In this scheme, TagGen accepts the integer probability parameter $0 < x < n$, which represents the number of nonzero components in the tag vector:

1. $\vec{t}$ is randomly selected from the set of all vectors with x nonzero entries.

2. t is computed as $\mathsf{Enc}_{\mathsf{IPE}}(m_{\mathsf{success}}, \vec{t}, \mathsf{pk})$.

Trials

1. $m \leftarrow \mathsf{Dec}_{\mathsf{IPE}}(\mathsf{t}, \mathsf{uk}, \mathsf{pk})$.

2. Return 1 if $m = m_{\mathsf{success}}$.

3. Otherwise, return 0.

5.3 Security

Theorem 1. *The IPE-based BBT scheme is secure with respect to the leakage function $\mathscr{L}(\mathsf{t}) = \mathsf{pk}$.*

Proof. The proof is simulation-based. For all PPT adversaries $\mathscr{A}(1^\lambda, \mathsf{uk}_1, \mathsf{uk}_2, \dots, \mathsf{uk}_n, \mathsf{t}, h(1^\lambda, \vec{t}))) = f(1^\lambda, \vec{t})$ there exists a PPT simulator that achieves the same advantage using only the trial results and the public key:

$$\mathscr{B}(1^\lambda, \mathsf{uk}_1, \mathsf{uk}_2, \dots, \mathsf{uk}_n, \mathsf{Trial}(\mathsf{uk}_1, \mathsf{t}), \mathsf{Trial}(\mathsf{uk}_2, \mathsf{t}), \dots, \mathsf{Trial}(\mathsf{uk}_n, \mathsf{t}), \mathsf{pk}, h(1^\lambda, \vec{t})) = f(1^\lambda, \vec{t})$$

The simulator $\mathscr{B}$ produces an output that is computationally indistinguishable from that of $\mathscr{A}$. The algorithm for $\mathscr{B}$ proceeds as follows:

```
$\vec{s} \leftarrow \langle 1, 1, \dots, 1, 1 \rangle$
for $1 \leq j \leq n$ do
    $\vec{v}_j \leftarrow \langle 0, 0, \dots, 1, \dots, 0, 0 \rangle$ where only the $j$th element is 1.
    $\mathsf{t}_j \leftarrow \mathsf{Encrypt}(\vec{v}_j, \mathsf{pk})$
end for
for all $\mathsf{uk}_i$ do
    if $\mathsf{Trial}(\mathsf{uk}_i, \mathsf{t})$ is success then
        for $1 \leq j \leq n$ do
            if $\mathsf{Trial}(\mathsf{uk}_i, \mathsf{t}_j)$ is not success then
                $\vec{s}_j \leftarrow 0$
            end if
        end for
    end if
end for
$\mathsf{s} \leftarrow \mathsf{Encrypt}(\vec{s}, \mathsf{pk})$
Run $\mathscr{A}(1^\lambda, \mathsf{uk}_1, \mathsf{uk}_2, \dots, \mathsf{uk}_n, \mathsf{s}, h(1^\lambda, \vec{t})))$ and output the result.
```

The output of algorithm $\mathscr{B}$ described above must be computationally indistinguishable from the output of $\mathscr{A}$; otherwise, an adversary could leverage the difference in the two to break the security of the underlying IPE scheme as follows:

1. Choose $\vec{t}$ as an arbitrary vector.

2. Submit arbitrary key vectors $\vec{k}_1, \vec{k}_2, \ldots, \vec{k}_n$.

3. Choose $\vec{s}$ as it would be computed by $\mathscr{B}$ (i.e., the vector with the maximal number of non-zero entries that is still orthogonal to all $\vec{k}_i$ orthogonal to $\vec{t}$).

4. Choose plaintext $m = m_{\mathsf{success}}$.

5. Submit challenge attribute vector $(\vec{t}, \vec{s})$ and challenge plaintext (m, m) and receive x, which is t if $b = 0$ or s if $b = 1$. Note that these submissions are admissible under the security definition of IPE because $\vec{s}$ and $\vec{t}$ are specifically constructed such that $\vec{s} \cdot \vec{k}_i = \vec{t} \cdot \vec{k}_i$ for all $\vec{k}_i$, as is required when $m_0 = m_1$.

6. Compute $\mathsf{out}_{\mathscr{A}} \leftarrow \mathscr{A}(1^\lambda, \mathsf{uk}_1, \mathsf{uk}_2, \ldots, \mathsf{uk}_n, \mathsf{x}, h(1^\lambda, \vec{t})))$.

7. If $\mathsf{out}_{\mathscr{A}}$ is as $\mathscr{A}(1^\lambda, \mathsf{uk}_1, \mathsf{uk}_2, \ldots, \mathsf{uk}_n, \mathsf{t}, h(1^\lambda, \vec{t})))$, output 0.

8. Otherwise, if the output $\mathsf{out}_{\mathscr{A}}$ is as $\mathscr{A}(1^\lambda, \mathsf{uk}_1, \mathsf{uk}_2, \ldots, \mathsf{uk}_n, \mathsf{s}, h(1^\lambda, \vec{t})))$, output 1.

Clearly, if the adversary has non-negligible advantage in distinguishing the outputs of $\mathscr{A}(1^\lambda, \mathsf{uk}_1, \mathsf{uk}_2, \ldots, \mathsf{uk}_n, \mathsf{t}, h(1^\lambda, \vec{t})))$ (which is exactly the output of the adversary $\mathscr{A}$) and $\mathscr{A}(1^\lambda, \mathsf{uk}_1, \mathsf{uk}_2, \ldots, \mathsf{uk}_n, \mathsf{s}, h(1^\lambda, \vec{t})))$ (which is exactly the output of the simulator $\mathscr{B}$), then the adversary also wins the IPE security game with non-negligible advantage. But, for a secure IPE scheme no such adversary can exist. Thus no PPT algorithm exists that can distinguish the output of the simulator $\mathscr{B}$ from the output of the $\mathscr{A}$.

□

5.4 Choice of Parameters

Besides the choice of underlying IPE scheme, the IPE-based BBT scheme also allows the choice of parameters for the dimension of the vector space n and the number of nonzero components a in each user key. Choices of these parameters will affect the number users that the system can support as well as the available choices for tag probabilities.

The IPE construction uses the number of nonzero components in a tag vector to control the probability of a successful trial. Therefore, for a system of dimension n there are n discrete probability "tiers" where tags in the ith tier have i nonzero components. Given an IPE scheme of dimension n, a nonzero components in each user key, and x nonzero components in a tag, the odds of a successful trial are:

$$\Pr[\text{success}] = \binom{n-x}{a} / \binom{n}{a}$$

Here, the numerator counts the number of ways to choose a user key that is orthogonal to the tag, and the denominator represents the total number of user keys possible.

This means that the probability tiers are not distributed uniformly. There are more tiers with lower probabilities of success than there are tiers with higher probabilities of success. For applications that require higher probabilities, we can simply invert the result of all trials to get a more favorable distribution. The remainder of this section follows this convention of inverting trial results. As a concrete example, figure 1 visualizes the case where $n = 64$ components are used.

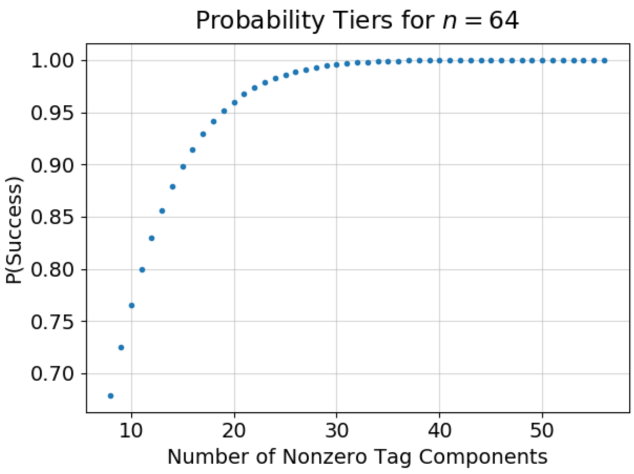


Figure 1: The distribution of probability tiers is biased toward the upper end of [0, 1] (using inverted trial results with $n = 64$, $a = 8$).

Two user keys uk_1 and uk_2 are called functionally unique if there exists a tag t such that $\mathsf{Trial}(\mathsf{uk}_1, \mathsf{t}) \neq \mathsf{Trial}(\mathsf{uk}_2, \mathsf{t})$. In other words, at least one of the associated key vectors has at least one non-zero component that is zero in the other vector, so that it is possible to construct a vector that is orthogonal to one but not the other. The number of functionally unique user keys depends on the number of components n and the choice of number of non-zero components in each user key a and is given simply as:

$$\binom{n}{a} = \frac{n!}{a!(n-a)!}$$

Table 5.4 compares the tag size in bits for IPE-BBT and the alternative scheme described in section 3. For the underlying IPE scheme, we used the state-of-the-art attribute-hiding IPE scheme due to Chen et al. [10] (our implementation using this scheme is discussed further in section 7). We assume a 1024-bit prime is used, for security equivalent to a symmetric key of 112 bits [16]. Chen's IPE scheme requires $4n + 4$ group elements for a ciphertext in an n-dimensional IPE scheme. We assume that group elements can be represented compactly by specifying only the x coordinate plus one bit indicating the y coordinate [25]. Thus the total size

Dimension	Users	IPE Size	ElGamal Size
9	9	5.1 KB	0.5 KB
10	45	5.6 KB	2.5 KB
11	165	6.2 KB	9.3 KB
12	495	6.7 KB	27.8 KB
16	12870	8.7 KB	723.9 KB
32	1.1×10^7	16.9 KB	591.7 MB
64	4.4×10^9	33.3 KB	249.0 GB

Table 1: Sizes of tags in a system supporting a given number of users in IPE-based BBT using Chen's IPE scheme, compared with semantically secure cipher construction using ECC ElGamal variant.

of a tag using Chen's IPE scheme is $(1024+1)(4n+4)$ for IPE dimension n.

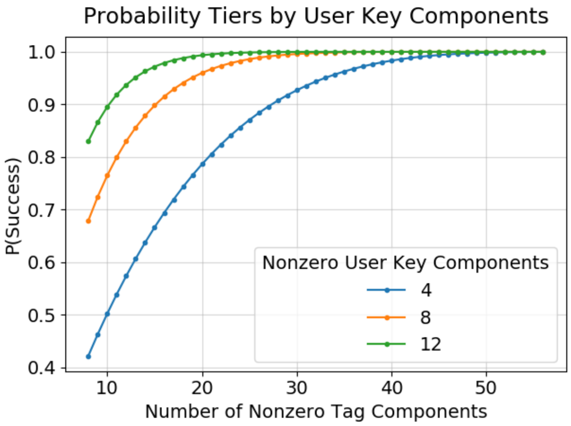
For comparison, we selected a public-key cryptosystem that represents minimal realistic storage requirements for a public key scheme at a comparable security level. We instantiated the semantically-secure encryption with an ECC variant of the ElGamal cryptosystem, which has been proven secure under elliptic curve discrete log assumptions [20]. This cryptosystem requires 2 group elements per ciphertext. We assume a 224-bit curve for a comparable level of security with the IPE scheme, again equivalent to a symmetric key strength of 112 [4]. This requires a total of 450 bits per ciphertext. The IPE scheme that supports 165 users (11 components with 8 non-zero user key components) uses less space than the corresponding public key scheme.

The number of possible tags is defined in the same way as it is for user keys. Two tags t_1 and t_2 are functionally unique if there exists a user key uk such that $\mathsf{Trial}(\mathsf{uk}, \mathsf{t}_1) \neq \mathsf{Trial}(\mathsf{uk}, \mathsf{t}_2)$. The number of functionally unique tags is different at each probability tier and depends on the total number of components in the vector space n and the number of nonzero components x used for that probability tier:

$$\binom{n}{x}$$

Therefore, it may be desirable to restrict the minimum and maximum probability tiers used so that the number of functionally unique tags at any probability tier does not fall below a chosen minimum. This means that the number of practically usable probability tiers may be less than n. For example, if one uses $n = 64$ components then one may only use tags with at least 8 components and no more than 56, which ensures that the number of functionally unique tags in any probability tier is at least $\binom{64}{8} \approx 2^{32}$.

The number of nonzero components a in each user key also affects the range and number of probability tiers: lower values of a allow a wider range of probability tiers, but fewer functionally unique user keys. Individual applications will need to determine a suitable trade-off. Figure 5.4 visualizes the available probability tiers and number of user keys by the number of nonzero user key components.



Nonzero Components	Users (to nearest power of 2)
4	2^{19}
8	2^{32}
12	2^{41}

Figure 2: Comparison of available probability tiers with IPE dimension $n = 64$ with various values of a (nonzero user key components). Lower values of a give greater flexibility in probability choice but support fewer users.

6 Practical Security

In practice, the security of any BBT scheme will require that an adversary does not have access to too many keys. With enough keys, it is possible for an adversary to approximate the trial probability. As with any distributed system, the security of BBT in a system will break down if an adversary compromises enough nodes. In this section we first consider ways an adversary might attempt to compromise a system and then develop a model that quantifies the amount of information an attacker learns about trials based on the number of compromised nodes.

One of the most obvious way an adversary may attempt to compromise a system is a Sybil attack. In a Sybil attack, a single adversary creates multiple fake identities and appears to the network as many users instead of one [14]. Fortunately, there are wide range of known defenses against Sybil attacks for different domains. The authority can attempt to manually attempt to verify node identities before issuing user keys. In cases where this is impractical, automated defenses exist. Social network-based defenses such as SybilGuard [38], SybilLimit [37], and SybilInfer [13] are capable of detecting Sybil nodes using the social relationships between nodes in the network, under the assumption that attackers are unable to create many trust relationships with legitimate users. Behavior-based schemes seek to distinguish between real and Sybil nodes via behaviors such as network activity and movement. For example, both work by Abbas et al. [1] and Jan et al. [17] utilize the heterogeneity of radio signals to detect Sybil attackers in MANETs and Wireless Sensor Networks respectively. Puzzle-based defenses such as SybilControl [22] utilize a proof of work based approach to mitigating Sybil attacks. Lastly, new approaches which leverage smart contracts, such as that proposed by Bochem et al. [7] put an economic price on Sybil identities.

Another simple attack is possible if a protocol misuses BBTs. For example, if a protocol requires multiple trials at the same probability protocol, then even a single user may be able to gain significant information about the probability of the associated tags. If a user observes several tags and knows (either from protocol specification or otherwise) that the tags all use the same probability parameter, then the user may use the differing results from the tags to approximate their shared probability.

In the event that an attacker does obtain multiple trial results, it is critical that we understand how much can be learned. The following subsections analyze the amount of knowledge that an attacker gains from multiple trial results, in both the ideal and the practical IPE schemes. Whether or not this amount of information leakage is considered acceptable is ultimately application-dependent.

Attacks on the Ideal Scheme

In an ideal BBT construction, the adversary learns only the trial results corresponding to the keys that it holds. If the adversary with n keys has no auxiliary information about the underlying distribution of success probabilities, then its best estimate for the success probability of a tag is $\frac{x}{n}$ where x is the number of observed successes. A confidence interval can also be computed to measure the uncertainty in this estimate. As an example, Figure 3 shows the upper and lower bounds of a 95% confidence interval on a tag with $p = 0.5$ as the number of the adversary's keys increases. The confidence interval is computed assuming that the adversary's trial results approximate the true tag probability as closely as possible, which is the best possible case for an attacker. We use a normal approximation to compute the confidence interval. Figure 3 shows that the attacker's knowledge increases with more trials. The attacker gains information rapidly at first, but trials beyond the first 10-15 bring diminishing returns. After 100 trials, the adversary is 95% confident that $0.4 < p < 0.6$.

However, if the adversary has a priori information on the underlying distribution of success probabilities, then the calculation is different. For example, if an adversary knows that all tags are drawn from discrete probability tiers (as in the case of the IPE-based scheme), then the adversary can use Bayesian inference to compute the likelihood that a tag comes from any tier given the a priori knowledge and the trial outcomes. Bayes' theorem gives the likelihood that a tag is from tier T given trial results R as:

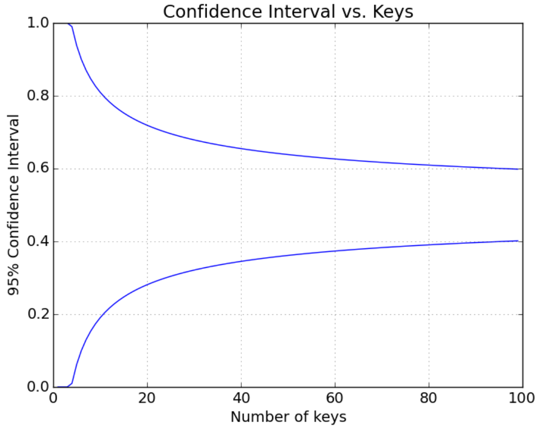


Figure 3: The upper and lower bounds of a 95% confidence interval for the probability of a tag with $p = 0.5$, as computed by an attacker whose trial results approximate a 50% success rate as closely as possible.

$$P(T|R) = \frac{P(T)P(R|T)}{P(R)} \tag{3}$$

where $P(T)$ is the prior likelihood of a probability tier T, $P(R|T)$ can be modeled as a binomial distribution, and $P(R)$ can be computed as $\sum P(R|T_i)P(T_i)$ for each probability tier T_i. This represents, from the adversary's point of view, the likelihood that a tag comes from a given probability tier given the observed trial results. This serves as an effective measure of what the adversary knows about the probability of a trial associated with the tag.

Figure 4 shows the expected view of an attacker in the ideal discrete case for two different tags in a simplified model that includes only 4 probability tiers. The tiers used are selected at roughly equal intervals from the tiers available in the IPE scheme with $n = 64$, and are listed in table 2. The attacker's confidence in each probability tier was computed using the Bayesian model outlined above and taken as an average over all possible attacker trial results (weighted using the binomial distribution for the likelihood of each result). We assume that each probability tier is equally likely to an attacker as a prior likelihood. As the number of trial results available to the attacker increases, the confidence in the true probability tier increases while the confidence in other tiers decreases.

Attacks on the IPE Scheme

We know that with ideal security, only the trial results are learned. In the IPE scheme, additional information is leaked (constrained by leakage function in the security proof). We

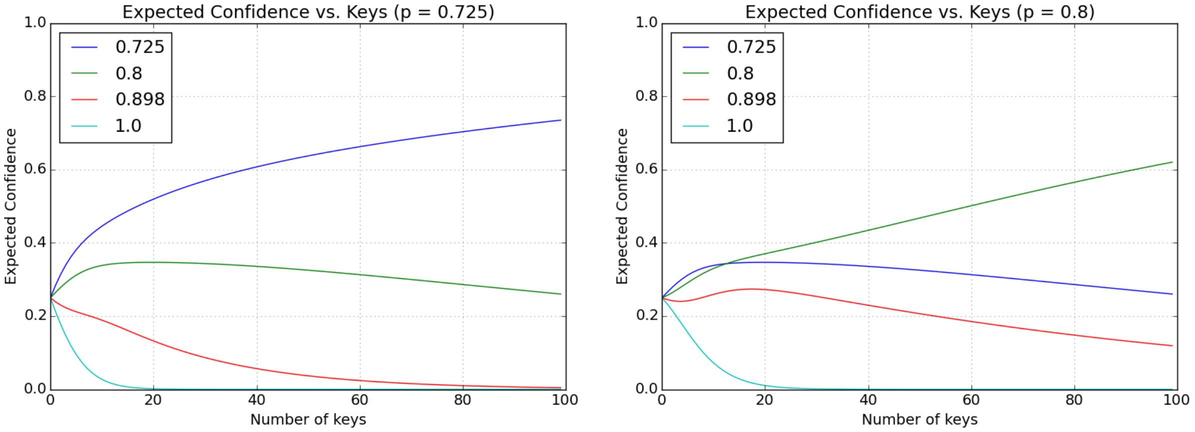


Figure 4: An attacker's view of two tags in the ideal discrete case, using a simplified model with 4 probability tiers. Each graph shows the attacker's expected confidence in each probability tier as a function of the number of trial results available to the attacker. The left graph shows the case where the true probability of the tag is $p = 0.725$, while the right graph shows a tag with $p = 0.8$.

Nonzero Components	p	Prior Likelihood
9	0.725	0.25
11	0.800	0.25
16	0.898	0.25
56	1.000	0.25

Table 2: Distribution of tags used in analysis.

wish to quantify how much an attacker can learn from the trial results, and how much more can be learned from the leaked information in the IPE scheme.

Essentially, the attacker can use information about the location of nonzero components in user keys to narrow down the set of possible tags. Knowing the components in each user key, together with their trial results for a tag, allows the attacker to quickly rule out any tag configurations that are inconsistent with the observed trial results.

The leakage function of the security proof takes this into account by allowing the public key of the underlying IPE scheme to leak. Since the public key is intentionally public in an IPE scheme, clearly this does not break the security guarantees of the underlying IPE; however, it does allow an adversary to potentially learn more about tags than the ideal case. Recall that the IPE public key allows one to encrypt a message under an arbitrary vector and obtain the ciphertext. In a BBT scheme, the IPE ciphertexts corresponds to BBT tags. Therefore, an adversary with the IPE public key can generate arbitrary tags, which allows the adversary to test user keys (but not tags) for the presence of any non-zero components. By repeated testing an adversary can determine exactly which components are zero and nonzero in each user key. In the rest of this analysis we assume the worst-case; that is, that the adversary already has access to the upper bound of information allowed by our security proof.

If an adversary knows which components in each user key are nonzero, then it can narrow the set of possible tags to those that give the same trial results for the same keys. Now, the adversary can estimate $P(R|T)$ as the proportion of possible tags from a tier that produce the same results when combined with same set of keys. For example, if trial result of testing a tag with one user key indicates that the two vectors are orthogonal, then any tags that share a nonzero component with the key are eliminated as possibilities. The attacker can count the number of consistent tags at each probability tier, and divide by the total number of possible tags in that tier to obtain a better estimate of $P(R|T)$. Again, $P(R)$ can be computed as $\sum P(R|T_i)P(T_i)$. The adversary can then again compute the overall likelihood that a tag comes from a given probability tier using the Bayesian inference described by equation 3.

Comparison

In order to determine the true impact of this attack, we compare the security of the IPE scheme to the ideal case by modelling two adversaries that each calculate the likelihood of tags differently. The component-aware adversary uses the full knowledge of the user keys components to compute the exact number of tags in each probability tier that could have produced the observed trial results, and then combines this with the prior likelihood of each probability tier to produce a confidence that a given tag comes from a given tier. Remember that no PPT adversary could hope to further distinguish between possible tags that would have produced the same trial results, since this directly contradicts the IPE security definition.

On the other hand, the naive adversary uses only the number of success and number of failures to compute the likelihood of probability tiers. The likelihood of an observed result given a probability tier is modeled only as a binomial distribution. This is the best that an adversary could hope to do under the ideal security definition, where only trial results are revealed.

Figures 5 and 6 show a comparison of the two attacks in one case. For simulating the two attacks we chose parameters that provide a reasonable balance of performance, security, and number of users supported: $n = 64$ as the dimension of the IPE scheme and $a = 8$ for the number of nonzero components per user key. For simplicity, we limited tags to only a set of a few that provided roughly evenly-spaced probability tiers from about 0.72 to 1.0, in 0.10 intervals. Table 2 lists the exact tags used. For the prior distribution of tags, each probability tier was assumed to be equally likely.

For each attack, we sampled a given number of random keys, ran the attack with the keys on a randomly sampled tag at each probability tier, and then reported the resulting computed distribution of tag likelihood for each tag. We repeated this process many times to obtain an average over uniformly random sampled n keys and tags sampled randomly from our distribution of probability tiers. Shannon entropy, defined as $-\sum p_i \log(p_i)$ can be used as a measure of the uncertainty over a distribution [33]. After each sampled attack, we computed the entropy of the computed probability tier distribution. We then computed the average expected entropy over the sampled sets of user keys and tags and graphed it as a function of number of user keys held by an attacker. Figure 5 shows that the difference in the attacker uncertainty is minimal between the ideal and IPE schemes. The expected entropy in a tag distribution from an attacker's point of view diminishes with each additional key known, and it diminishes slightly faster for the IPE-based scheme than it does in an ideal scenario.

We also computed the expected Kullback-Leibler divergence between the component-aware model and the naive model, using the same tag distribution and random sampling method. The Kullback-Leibler divergence between two distributions P and Q is defined as $-\sum p_i \log(\frac{p_i}{q_i})$. This divergence measures the amount of information that an attacker gains about a tag probability by exploiting the information leakage in an IPE-based BBT scheme. Figure 6 shows that the additional information leaked to an attacker is minimal and quickly levels off around 0.02 bits.

7 Evaluation

We implemented IPE-based BBT using the adaptively attribute-hiding IPE scheme by Chen, Gong, and Wee, which is the current state of the art [10]. Chen et al. propose multiple variations with different performance characteris-

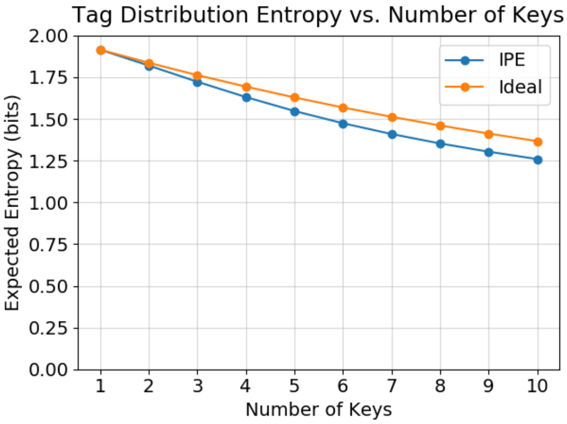


Figure 5: Expected entropy of computed tag likelihood as a function of attacker's number of keys.

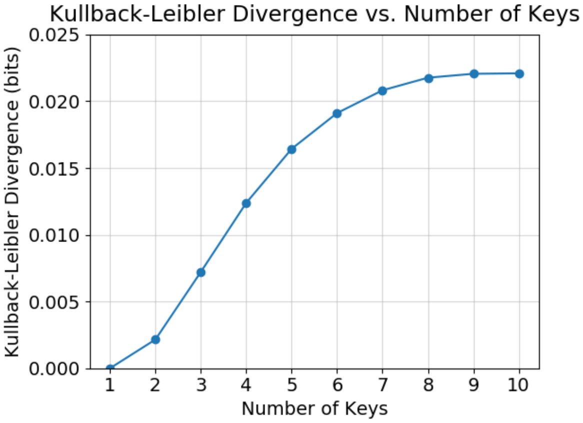


Figure 6: Expected Kullback-Leibler divergence between the component-aware model and the naive model as a function of attacker's number of keys.

Object	Size (in elements)
Public key	$8n+16$
User key	7
Tag	$4n+4$

Table 3: Size of objects in IPE-BBT implemented with IPE scheme from Chen et al., in number of group elements.

tics using different standard assumptions. We implemented the variant described in Section 4.4 of their paper, which is proved secure under the external decisional linear assumption.

Under this scheme, tags and the public key both require space that is $O(n)$ in the number of dimensions, or $O(\log(l))$ in the number of functionally unique user keys. User keys have a constant space requirement of 7 group elements. Table 3 details the exact space requirements for each object.

Setup, trials, user key generation, and tag generation all run in $O(n)$ time. For a typical number of dimensions, the trial run time is dominated by the pairing operations. Crucially, trials in this scheme require only 7 pairing operations, regardless of the number of dimensions.

We tested the speed of this implementation on a single core of an Intel Xeon E5-2680 v4 CPU clocked at 2.40GHz. For pairing operations, we used the Stanford Pairing-Based Cryptography (PBC) library [23, 24]. The curve used was of PBC's "Type A," which are curves of the form $y^2 = x^3 + x$ over the field $\mathbb{F}_q$, where q is a prime such that $q = 3 \bmod 4$. q was chosen as a random 1024-bit prime, and the parameter r was chosen as a 224-bit number. Since the curve has embedding degree $k = 2$, these parameters are equivalent to the strength of an 112-bit symmetric key, according to the IEEE Standards for pairing-based cryptography [16].

Although the performance of all BBT steps must be reasonable, the Trial step is of the most concern. Trials are expected to be carried out by clients who may have limited resources, such as mobile devices. In contrast, Setup, TagGen, and KeyGen are all expected to be carried out by the single authority which would likely have access to significantly more resources.

Figure 7 shows the runtime of each BBT algorithm in our implementation using Chen, Gong, and Wee's IPE scheme. As expected, each algorithm shows a clear linear trend as the dimension is increased. As previously mentioned, the performance of the trial algorithm is the most critical, since disconnected clients with limited computing resources will be running it. Our results show that the trial algorithm is quite practical with parameters that can support a large number of users. For example, with $n = 64$ dimensions and $a = 8$ nonzero components per user key, there are approximately 2^{32} functionally unique user keys and a trial takes about 29 ms.

8 Applications

Any system that requires participants to take actions probabilistically can use BBTs to enhance privacy. Specifically, we envision several possible applications for BBTS in secure distributed systems. We provide two example scenarios that could benefit from deployment of blinded Bernoulli trials.

Probabilistic Forwarding of Content in a Network

Some networks (especially peer-to-peer networks) employ random walks based on probabilistic forwarding of content for privacy reasons. For example, in the anonymous communication systems such as AP3 [26] and DiscountANODR [36], when presented with a message, nodes randomly decide to either forward messages to another node in the mix or to send the message directly to its destination. The random process of forwarding obscures the origin of a message: when a node receives a message, it does not know if the message originated from the immediate preceding node or if it comes from 1, 2, or more hops away from that node. The use of the random coin rather than full circuit specification relieves the sending node from having to maintain state about the network topology outside of its immediate neighbors. The network uses a parameter p to specify the probability that nodes forward messages on to another node.

This approach introduces a trade-off between anonymity and network overhead: for lower values of p, messages take shorter paths (on average) through the network, but an observer can narrow down the set of likely originators to a smaller set based on the overlay distance to mix nodes and the distribution of random walk lengths. Higher values of p increase the number of possible nodes that might be the originator, but reduce network performance due to longer random walks. The authors of AP3 propose p between 0.5 and 0.9.

Using BBTs we can construct a network that allows for differential service, providing some users faster traffic, while still retaining the anonymity of longer paths. For example, consider a system with two classes of traffic. The Priority Class wants higher performance, and therefore shorter random walks, achievable with a low value of p. On the other hand, the Slow Class has no performance demands, so it can tolerate a higher value of p, increasing the size of its anonymity set. Without BBTs the traffic classes are trivial to distinguish, and as a result Priority traffic can be analyzed in a vacuum, leading to small anonymity sets. Marking a message's p with a BBT means that the two classes can not be distinguished based on this value, and in turn the faster class can benefit from the adversary's uncertainty about which peers should be included in the set of possible originators. This approach works especially well when the majority of the traffic falls into the Slow Class.

To evaluate the utility of BBT in this system, we simulated

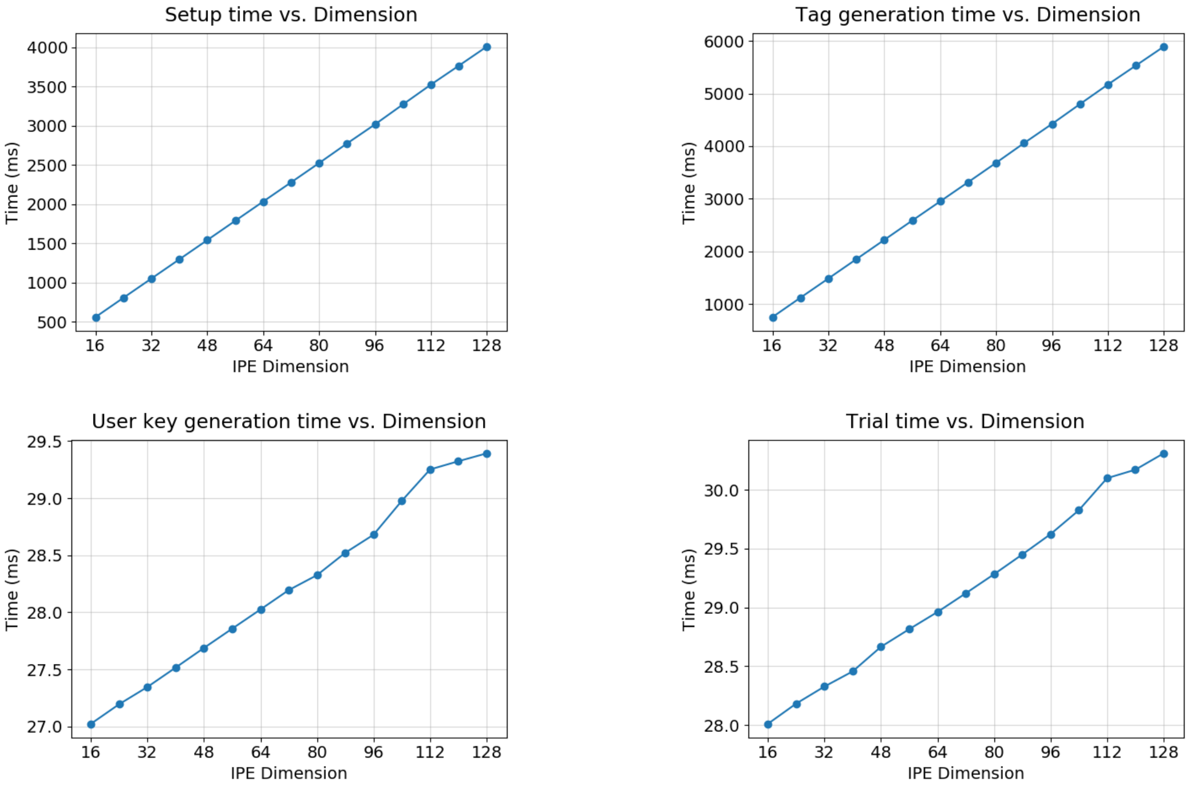


Figure 7: Runtimes for individual steps of IPE-based BBT implementation, by the dimension of the IPE.

a network that uses probabilistic forwarding with two traffic classes. Priority traffic uses $p = 0.5$, while Slow traffic in our test network uses $p = 0.9$. We simulated the operation of a network containing 1,000 nodes, with each node maintaining at least 4 connections to other nodes. The resulting graph had an average mixing time of slightly more than 7. In this network, Priority traffic represents 10% of total messages, and it is assumed that the adversary knows both the overall proportion of Priority traffic in the network and the full network topology. As expected, Figure 8 shows that Slow traffic takes drastically longer paths through the network, while Priority traffic reaches its destination much quicker. If an observer can distinguish Priority traffic, then this creates a privacy concern: because Priority traffic originates from nearby nodes with high probability, a node that receives it and recognizes it as Priority traffic has a high degree of confidence that the sender is in the small set of nodes that are nearby in the topology. However, if a BBT scheme is used to blind the priority class of traffic in the network, then Priority traffic can benefit from the reduced latency without resorting to unacceptably small anonymity sets. As Figure 9 illustrates, using BBT in the network increases the anonymity set sizes to levels that are near those of the Slow class. On average, anonymity sets for the merged class of traffic resulting from BBT blinding of priority is slightly lower than if only slow traffic is considered. This is because the adversary knows the relative frequency of fast and slow traffic, and can adjust computation of likely nodes based on this information.

Beyond the example systems of AP3 and DiscountANODR, many anonymous communication protocols feature a system parameter taking the form of a probability. Examples include Freenet [12], Crowds [35], and Imprecise Routing [11]. In each of these protocols, BBT can be used in a similar manner to adjust network behavior, allowing for different security properties while blending in with standard traffic.

Intrusion Detection in Wireless Sensor Networks

Another application for BBTs is masking how many nodes are conducting a specific behavior. As an example, consider a wireless sensor networks comprised of a large number of low-cost embedded devices conducting measurements in an environment [32]. They rely on short-range wireless communication between low-power devices (often battery-powered), which makes power consumption a top concern. Because of the communication range constraints and large number of devices over a wide area, direct connectivity is limited; instead sensors distribute messages from device to device over multiple hops in the wireless network.

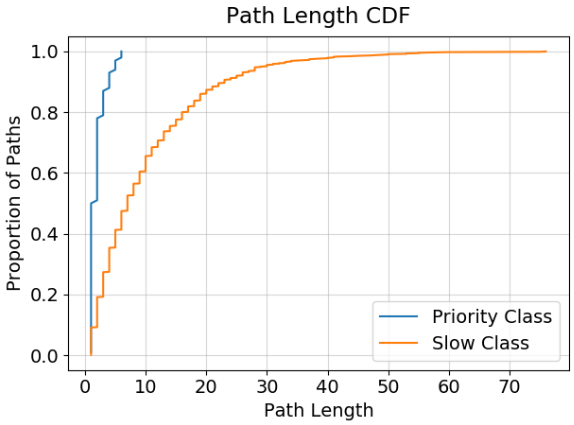


Figure 8: The distribution of path lengths for different traffic classes in the simulated network. Priority traffic takes significantly fewer hops to reach its destination, which translate to improved reliability and decreased latency.

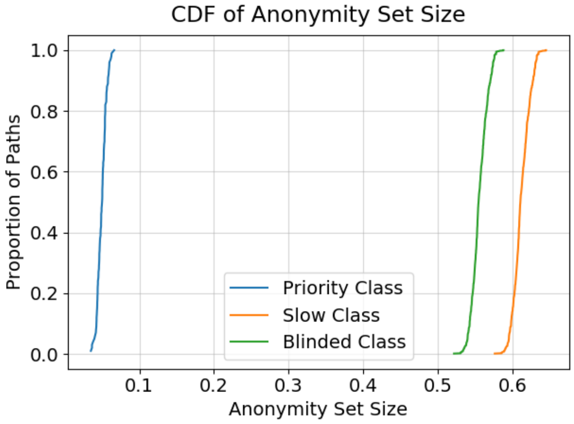


Figure 9: The distribution of anonymity set sizes for traffic classes. Without BBT, the anonymity set for nodes sending Priority traffic is less than 10% of the network. If the classes are indistinguishable, the anonymity set size is more than 50% of the network.

Sensor networks deployed in a hostile environment face the additional complication that certain compromised sensors may not be trustworthy. Adversaries may attempt to falsify sensor readings from compromised nodes. Recent work has examined methods for detecting compromised nodes in sensor networks [9, 34, 39]. In general, prior work has dealt with detecting falsified results by comparing them to a consensus of trustworthy results, under the assumption that an adversary is unable to compromise enough nodes to form a false consensus.

Of course, building a consensus requires a large number of measurements. Again, this leads to a trade-off between security and efficiency: employing many nodes in redundant measurements to build a consensus raises the bar for an attacker, but each sensor measurement uses energy (which is an extremely limited resource in wireless sensor networks). An obvious compromise would be to perform some fraction of measurements using only a small number of sensors, and sometimes use large-scale "audit" measurements to provide the necessary data for intrusion detection. If the metadata associated with a measurement reveals the number of nodes involved in the measurement, then this solution is vulnerable to an obvious attack: an adversary can simply refrain from lying during the audit measurements.

BBTs mitigate the aforementioned attack by limiting the adversary's ability to distinguish typical measurements from audit measurements. The authority can control the scope of the measurement using the probability of a tag, and nodes can decide their involvement in the measurement with a trial. This prevents the attacker from selectively influencing measurements without detection. In addition, the authority can easily verify that nodes are not returning results for spurious measurements by duplicating the deterministic trial results and verifying that the node originating the measurement was in fact one of the nodes directed to take the measurement.

We evaluated the effect of applying this technique on a small network of sensor nodes. We modeled the network as 100 independent sensor nodes, each capable of a fixed finite number of measurements before it is considered expired. The adversary controls one node. We assume that the malicious node is detected if it lies on a measurement that is performed by at least half of the nodes. For normal measurements, the network uses a tag with $p = 0.05$. For audit measurements, the network sets $p = 0.6$ (this ensures with high probability that at least half of the network does actually perform the measurement). We also assume that the authority verifies that each returned result is tied to an actual successful trial. This can be accomplished in general by requiring that nodes attach their user key to each result, encrypted so that only the authority can read it. Note that this is consistent with the threat model which already assumes that the authority has unlimited access to user keys.

By varying the "audit rate" (the fraction of measurements that are audit measurements), the authority can select an arbi-

Audit Rate	Probability of Detection	Relative Lifetime
0.0	0.00	12.0
0.2	0.71	3.7
0.4	0.86	2.2
0.6	0.93	1.6
0.8	0.97	1.2
1.0	0.99	1.0

Table 4: The trade-off between sensor lifetime and detection probability. Here, the audit rate is the proportion of measurements that are audit measurements; the detection probability is the probability that a single lie is detected; and the relative lifetime is the lifetime of the entire network relative to the base case where all measurements are audit measurements. Allowing a small probability of an undetected attack can significantly increase the lifetime of the network.

trary trade-off between efficiency (conserving resources for more useful measurements) and security (performing redundant measurements to detect attacks). As the audit rate increases, so does the probability of detecting a malicious measurement; on the other hand, as the audit rate decreases, the lifetime of the network increases. Table 4 shows this trade-off. We sampled audit rates and computed both the resulting sensor lifetime and the probability that the adversary is detected each time it lies. The expected sensor lifetime is reported relative to the lifetime in the "safe" network (that is, one with an audit rate of 1). Even at a relatively low audit rate of 0.2, the probability of detection is high (71%) from the perspective of the sensor. This is because the number of sensors employed for normal measurements is much smaller than the number of sensors involved in an audit measurement. As a result, a given sensor is more likely to be selected via a large audit measurement than it is to be selected for a normal measurement.

Because the adversary cannot distinguish normal and audit measurements, it cannot selectively lie. This effectively limits the number of times an adversary can lie without detection (with overwhelming probability). Without using BBT, the adversary is only prevented from lying during audit measurements; otherwise, it can forge measurements without detection indefinitely. With BBT, if the adversary has a 50% chance of detection per measurement (for example) then it can expect to lie only about twice before detection, on average.

Discussion

In practice, the bandwidth and computation overhead of a BBT scheme will determine its usefulness for any particular application. In the first application, we assume that the overhead of computing a trial is low relative to the work required for a message forward. This assumption is reasonable, for example, in AP3, where a single forward requires a distributed hash table lookup and therefore multiple round-trip messages with peers. In this scenario the combined latency of one forward can be significantly slower than a trial in the IPE-based BBT construction.

For the sensor network application, the energy savings of skipping measurements will have to be weighed against the cost of performing the trials at each sensor. Depending on resource availability, different BBT constructions might be more appropriate. For example, if bandwidth is cheap but computational resources are constrained, then the IND-CPA construction presented 3 might actually be more suitable.

9 Conclusion

Although many distributed systems make use of probabilistic actions, systems so far either specify the probability as a fixed parameter or reveal the varying probabilities to each user. Blind Bernoulli trials are a privacy-enhancing measure that preserves the semantics of Bernoulli trials across a set of nodes, while hiding the exact parameters from individuals.

Fundamentally, Blind Bernoulli trials reveal the trial outcome without revealing the trial parameters. We create a definition that formalizes the idea that users should learn "no more" about the trial parameters than they would by receiving only the trial results corresponding to the keys held, and explore some possible solutions that meet our proposed definition.

Since BBTs are a special case of functional encryption (FE), they can easily be implemented with any FE primitive that allows arbitrary functions. However, since practical general functional encryption is not currently available, there is a need for a specific scheme that achieves the same results with a more efficient algorithm. Existing forms of functional encryption for specific classes of functions can be used to instantiate much more practical Blind Bernoulli trials, albeit with some security loss. Specifically, we can construct a near-ideal BBT scheme from inner product encryption by varying the number of nonzero components in tags to control the probability of their trials.

We prove the near-ideal security of the IPE-based scheme under our definition by showing a reduction to the security of the underlying IPE scheme. This definition takes into account the security loss and places an upper bound on exactly how much information is revealed to an adversary, even one who controls multiple keys. By simulating the attacker's point of view on a large number of trials and keys, we can measure the average uncertainty towards the distribution of possible tags for an adversary with multiple keys: the expected entropy of a tag distribution decreases steadily with the number of trial results known. We compare the entropy loss in the ideal case to the IPE-based BBT scheme and show that the additional entropy loss in the IPE case is small.

Finally, we implement the IPE-based scheme in software and analyze its efficiency. We show that, in a system with realistic parameters, trials can be executed in a reasonable amount of time. Also, tags and keys in the IPE scheme are small (logarithmic in the number of users). Even with a relatively small number of users (on the order of 100), this is less storage than our simple semantically-secure cipher solution with linear keys.

We conclude that Blind Bernoulli trials can be efficiently implemented using IPE, and that they are an effective way obscure probability parameter metadata. This paper proposes two potential applications where this can prevent attacks that would otherwise exploit knowledge of the probability parameter. We hope that others in the security and distributed systems communities will explore additional uses for the primitive.

References

[1] Abbas, S., Merabti, M., Llewellyn-Jones, D., and Kifayat, K. Lightweight sybil attack detection in manets. *IEEE systems journal 7*, 2 (2013), 236–248.

[2] Agrawal, S., Agrawal, S., Badrinarayanan, S., Kumarasubramanian, A., Prabhakaran, M., and Sahai, A. Function Private Functional Encryption and Property Preserving Encryption - New Definitions and Positive Results. *IACR Cryptology ePrint Archive* (2013).

[3] Agrawal, S., Freeman, D. M., and Vaikuntanathan, V. Functional Encryption for Inner Product Predicates from Learning with Errors. In *Advances in Cryptology – EUROCRYPT 2010*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 21–40.

[4] Barker, E. SP 800-57 Part 1 Rev. 4: Recommendation for Key Management Part 1: General, Jan. 2016.

[5] Bishop, A., Jain, A., and Kowalczyk, L. Function-Hiding Inner Product Encryption. In *Advances in Cryptology – ASIACRYPT 2015*. Springer, Berlin, Heidelberg, Berlin, Heidelberg, Nov. 2015, pp. 470–491.

[6] Blum, M. Coin flipping by telephone a protocol for solving impossible problems. *ACM SIGACT News 15*, 1 (1983), 23–27.

[7] Bochem, A., Leiding, B., and Hogrefe, D. Unchained identities: Putting a price on sybil nodes in mobile ad hoc networks. *Security and Privacy in Communication Networks (SecureComm 2018). Singapore (August 2018)* (2018).

[8] Boneh, D., Sahai, A., and Waters, B. Functional Encryption - Definitions and Challenges. *TCC 6597*, Chapter 16 (2011), 253–273.

[9] Chatzigiannakis, V., Papavassiliou, S., Grammatikou, M., and Maglaris, B. Hierarchical anomaly detection in distributed large-scale sensor networks. In *Computers and Communications, 2006. ISCC'06. Proceedings. 11th IEEE Symposium on* (2006), IEEE, pp. 761–767.

[10] Chen, J., Gong, J., and Wee, H. Improved inner-product encryption with adaptive security and full attribute-hiding. In *International Conference on the Theory and Application of Cryptology and Information Security* (2018), Springer, pp. 673–702.

[11] Ciaccio, G. Improving sender anonymity in a structured overlay with imprecise routing. In *International Workshop on Privacy Enhancing Technologies* (2006), Springer, pp. 190–207.

[12] Clarke, I., Sandberg, O., Wiley, B., and Hong, T. W. Freenet - A Distributed Anonymous Information Storage and Retrieval System. *Workshop on Design Issues in Anonymity and Unobservability* (2000).

[13] Danezis, G., and Mittal, P. Sybilinfer: Detecting sybil nodes using social networks. In *NDSS* (2009), San Diego, CA, pp. 1–15.

[14] Douceur, J. R. The Sybil Attack. *IPTPS* (2002).

[15] Goldwasser, S., Kalai, Y., Popa, R. A., Vaikuntanathan, V., and Zeldovich, N. Reusable garbled circuits and succinct functional encryption. In *the 45th annual ACM symposium* (New York, New York, USA, 2013), ACM Press, pp. 555–564.

[16] 1363.3-2013 - IEEE Standard for Identity-Based Cryptographic Techniques using Pairings. IEEE, 2013.

[17] Jan, M. A., Nanda, P., He, X., and Liu, R. P. A sybil attack detection scheme for a centralized clustering-based hierarchical network. In *Trustcom/BigDataSE/ISPA, 2015 IEEE* (2015), vol. 1, IEEE, pp. 318–325.

[18] Katz, J., Sahai, A., and Waters, B. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. *EUROCRYPT 4965*, Chapter 9 (2008), 146–162.

[19] Kawai, Y., and Takashima, K. Predicate- and Attribute-Hiding Inner Product Encryption in a Public Key Setting. *Pairing 8365*, Chapter 7 (2013), 113–130.

[20] Koblitz, N. Elliptic Curve Cryptosystems. *Mathematics of Computation 48*, 177 (Jan. 1987), 203–209.

[21] Lewko, A., Okamoto, T., Sahai, A., Takashima, K., and Waters, B. Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In *Advances in Cryptology – EUROCRYPT 2010*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 62–91.

[22] Li, F., Mittal, P., Caesar, M., and Borisov, N. Sybilcontrol: Practical sybil defense with computational puzzles. In *Proceedings of the seventh ACM workshop on Scalable trusted computing* (2012), ACM, pp. 67–78.

[23] Lynn, B. Stanford Pairing-Based Cryptography Library. `https://crypto.stanford.edu/pbc/`, 2006–2013.

[24] Lynn, B. On the implementation of pairing-based cryptosystems. Dissertation, 2007. `https://crypto.stanford.edu/pbc/thesis.pdf`.

[25] Menezes, A., and Vanstone, S. A. Elliptic Curve Cryptosystems and Their Implementations. *Journal of Cryptology* (1993).

[26] Mislove, A., Oberoi, G., Post, A., Reis, C., Druschel, P., and Wallach, D. S. Ap3: Cooperative, decentralized anonymous communication. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop* (2004), ACM, p. 30.

[27] Okamoto, T., and Takashima, K. Achieving Short Ciphertexts or Short Secret-Keys for Adaptively Secure General Inner-Product Encryption. In *Advances in Cryptology – EUROCRYPT 2010*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 138–159.

[28] Okamoto, T., and Takashima, K. Adaptively attribute-hiding (hierarchical) inner product encryption. EUROCRYPT 2012, 2012. `https://eprint.iacr.org/2011/543`.

[29] Okamoto, T., and Takashima, K. Fully Secure Unbounded Inner-Product and Attribute-Based Encryption. *ASIACRYPT 7658*, Chapter 22 (2012), 349–366.

[30] O'Neill, A. Definitional Issues in Functional Encryption. *IACR Cryptology ePrint Archive* (2010).

[31] Pinkas, B., Schneider, T., Smart, N. P., and Williams, S. C. Secure two-party computation is practical. In *International Conference on the Theory and Application of Cryptology and Information Security* (2009), Springer, pp. 250–267.

[32] Pottie, G. J., and Kaiser, W. J. Wireless integrated network sensors. *Communications of the ACM 43*, 5 (2000), 51–58.

[33] Rènyi, A. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics* (Berkeley, Calif., 1961), University of California Press, pp. 547–561.

[34] Sheng, B., Li, Q., Mao, W., and Jin, W. Outlier detection in sensor networks. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing* (2007), ACM, pp. 219–228.

[35] Shields, C., and Levine, B. N. A protocol for anonymous communication over the internet. In *Proceedings of the 7th ACM conference on Computer and communications security* (2000), ACM, pp. 33–42.

[36] Yang, L., Jakobsson, M., and Wetzel, S. Discount anonymous on demand routing for mobile ad hoc networks. In *Securecomm and Workshops, 2006* (2006), IEEE, pp. 1–10.

[37] Yu, H., Gibbons, P. B., Kaminsky, M., and Xiao, F. Sybillimit: A near-optimal social network defense against sybil attacks. In *2008 IEEE Symposium on Security and Privacy (sp 2008)* (2008), IEEE, pp. 3–17.

[38] Yu, H., Kaminsky, M., Gibbons, P. B., and Flaxman, A. D. SybilGuard - defending against sybil attacks via social networks. *IEEE/ACM Trans. Netw.* (2008).

[39] Zhang, K., Shi, S., Gao, H., and Li, J. Unsupervised outlier detection in sensor networks using aggregation tree. In *International Conference on Advanced Data Mining and Applications* (2007), Springer, pp. 158–169.