TRANSACTIONS ON DATA PRIVACY 2 (2009) 131–139

Cryptographically Secure Bloom-Filters

Ryo Nojima*, **Youki Kadobayashi****

*National Institute of Information and Communications Technology (NICT), 4-2-1 Nukuikitamachi, Koganei, Tokyo, 184-8795, Japan.

** Nara Institute of Science of Technology (NAIST), 8916-5 Takayamacho, Ikoma, Nara, 630-0192, Japan.

E-mail: ryo-no@nict.go.jp

Abstract. In this paper, we propose a privacy-preserving variant of Bloom-filters. The Bloom-filter has many applications such as hash-based IP-traceback systems and Web cache sharing. In some of those applications, equipping the Bloom-filter with the privacy-preserving mechanism is crucial for the deployment.

In this paper, we propose a *cryptographically secure* privacy-preserving Bloom-filter protocol. We propose such two protocols based on blind signatures and oblivious pseudorandom functions, respectively. To show that the proposed protocols are secure, we provide a reasonable security definition and prove the security.

1 Introduction

The Bloom filter provides us an efficient test for the group membership [4]. More precisely, let $S \subseteq U$ be a set and let $x \in U$ be an element, where U is some universe set. Then by utilizing the Bloom filter we can test whether or not $x \in S$ in the space efficient way. The Bloom filter has various applications in practice. For example, these applications include a Web-cache sharing and a hash-based IP-traceback [6].

In this paper, we consider the *privacy-preserving* variant of the Bloom filters. Taking the privacy into account is important in some of the applications of the Bloom filters. What we need to solve in those applications can be simplified as follows. There are two parties, named a server and a client, who have a set S and an element x, respectively. Here, we assume S is stored in the Bloom filter, and denote it by $\mathbf{BF}_S$. The client wants to verify whether or not $x \in S$ while preserving $\mathbf{BF}_S$ and x *secret* to each other. Ignoring the privacy, this problem can be solved easily. (That is, the server simply sends the entire Bloom filter $\mathbf{BF}_S$ to the client.) Also if the server has S instead of the filter $\mathbf{BF}_S$ then the problem can be solved by the secure set-intersection protocol [10]. But this protocol does not utilize the nice property of the Bloom filter. This is because, after transforming S into $\mathbf{BF}_S$, we can not obtain S information theoretically. Therefore, the server needs to store the original S instead of $\mathbf{BF}_S$ in this case. There are some cryptographic protocols which uses the Bloom-filters [1, 13]. Among those protocols, the one proposed by Bellovin and Cheswick [1] is the closest to ours. In [1], the authors proposed the searching scheme based on "encrypted" Bloom filters. We can regard their protocol as a privacy-preserving Bloom-filter protocol. However, in their protocol, there is a *semi-honest third party* and he/she plays an important role in protecting users' privacy. A cryptographic primitive proposed in [13] has many applications but obtaining the desired one without employing the trusted third party seems

difficult because it is similar in construction to [1]. On the other hand, in this paper, we avoid employing such a third party by using the cryptographic primitives known as the *blind signature schemes* (e.g., [8]) and the *oblivious pseudorandom functions* [9].

1.1 Intuition of the Protocol

As noted above, we propose an efficient privacy-preserving Bloom filter protocol based on the blind signature schemes and the oblivious pseudorandom functions. In the blind signature scheme, the server has a signing key sk, and the client has a message m and a verification key vk. The goal of the protocol is the client obtaining the digital signature of m under the signing key sk while keeping m and sk secret to each other. The oblivious pseudorandom function has similar aim. Let f_k be a pseudorandom function keyed by k. That is, there is no efficient algorithm which can distinguish f_k and the truly random function via black-box accessing. The oblivious pseudorandom function is a two-party (a server and a client) protocol. The input to the server and the client are k and m, respectively, and the goal of the protocol is the client obtaining $f_k(m)$ while k and m remain secret to each other.

To provide the intuition behind our proposal, we first introduce the Chaum's blind signature [8, 2], and then construct a *secure matching* protocol. Note that this protocol becomes the fundamental tool in our proposal. In the secure matching protocol, there are two parties, a sever and a client who have s and c, respectively. The goal of the protocol is the client verifying whether or not $c = s$ while keeping c and s secret to each other.

The Chaum's blind signature is defined as follows. Let $N := PQ$ be a large integer with P, Q primes of the same length. Also let e, d be integers such that $ed = 1 \bmod (P-1)(Q-1)$. Suppose $H : \{0,1\}^* \to \mathbb{Z}_N^*$ be a one-way hash function. We define the signing-key sk as $sk := (d, N, H)$ and the verification-key as $vk := (e, N, H)$. The signature of the message m is defined by $H(m)^d \bmod N$. So this is the FDH (Full Domain Hash) signature scheme [3]. To obtain the signature of the message m, the client chooses $r \in \mathbb{Z}_N^*$ randomly, computes and sends $r^e H(m) \bmod N$ to the server. The server computes and sends $(r^e H(m))^d \bmod N$ to the client. The client outputs

$$(r^e H(m))^d r^{-1} = H(m)^d \bmod N.$$

To construct the secure matching protocol, we need an additional hash function H' which is defined by $H' : \{0,1\}^* \times \mathbb{Z}_N^* \to \{0,1\}^\kappa$. The server first computes $H'(s, H(s)^d \bmod N)$ and sends it to the client. The client obtains $H(c)^d \bmod N$ by the blind signature scheme and checks

$$H'(s, H(s)^d \bmod N) = H'(c, H(c)^d \bmod N).$$

The protocol is summarized in Figure 1.

Intuitively, in [14, 7], this protocol is extended into the oblivious transfer and, in [12], the blind signature is replaced by the oblivious pseudorandom function and then the set-intersection protocol is obtained. In this paper, we extend this protocol to obtain the privacy-preserving Bloom-filter protocol.

1.2 Organization of This Paper

In Section 2, we begin with what the Bloom-filter is and the security requirements to the privacy-preserving Bloom-filter protocol are provided. In Section 3 and 4, we show our proposed protocols based on the blind signature schemes and oblivious pseudorandom

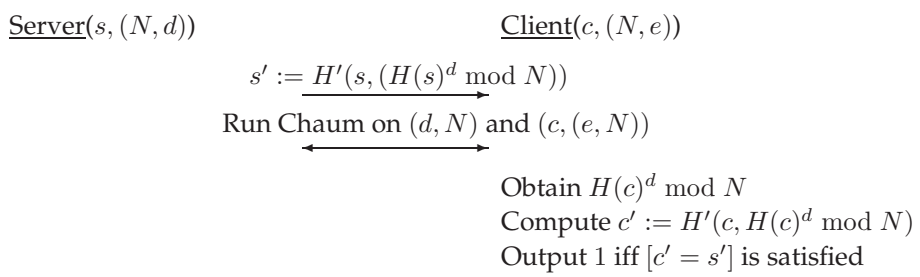

Figure 1: Privacy-preserving matching protocol based on the blind signature

functions, respectively. In both sections, we briefly introduce the blind signature scheme and the oblivious pseudorandom function. We conclude this paper with Section 5. This section will include a simple comparison (due to the black-box use of the blind signature scheme and the oblivious pseudorandom function) of two proposed protocols and our future research direction.

2 Preliminary

We define $[s] = 1$ if the statement s is true and 0 otherwise.

2.1 Bloom Filters

A Bloom filter [4] provides us an efficient test for the group membership. In this method, an m-bit array $\mathbf{BF}$ which represents a set $S = \{s_1, \ldots, s_n\}$ and l independent truly random hash functions

$$H_1, H_2, \ldots, H_l : \{0,1\}^* \to \{0, 1, \ldots, m-1\}$$

are employed. Initially, all bits in the array $\mathbf{BF}$ are set to 0. When we need to insert a word x to $\mathbf{BF}$, we compute

$$i_1 = H_1(x), \ldots, i_l = H_l(x)$$

and set

$$\mathbf{BF}[i_1] = 1, \ldots, \mathbf{BF}[i_l] = 1,$$

where $\mathbf{BF}[i-1]$ is an element of ith position in $\mathbf{BF}$. Here if $\mathbf{BF}[i]$ is already set to 1, we do nothing. We denote S embedded in Bloom filter by $\mathbf{BF}_S$.

To test, we just compute $i_1 = H_1(x), \ldots, i_l = H_l(x)$ and output 1 if and only if

$$[\mathbf{BF}[i_1] = 1] \wedge \cdots \wedge [\mathbf{BF}[i_l] = 1]\,.$$

Here we denote $x \in \mathbf{BF}_S$ if

$$[\mathbf{BF}[i_1] = 1] \wedge \cdots \wedge [\mathbf{BF}[i_l] = 1]$$

is satisfied.

In this (randomized) data structure if the set S is stored and $x \in S$ then $x \in \mathbf{BF}_S$. That is, the probability of the false negatives is 0. However, the probability of false positives is not 0. That is, there is a case that even if $x \in \mathbf{BF}_S$, $x \notin S$. However, mathematical analysis shows that for every $x \notin S$ the algorithm returning 1 is approximately $(1 - e^{-ln/m})^l$ which is small enough for the practical use.

2.2 Security Definition

In this paper, we consider the privacy variant of the Bloom filter protocol. In this protocol, there are two parties, a *server* and a *client*, who have $(S \subseteq U, m, l)$ and $(x \in U, n = |S|, m, l)$, respectively, where U is the universe set. The server stores S in the Bloom filter $\mathbf{BF}$ of size m with l hash functions, denoting $\mathbf{BF}_S$. And the goal is the client obtaining $[x \in \mathbf{BF}_S]$ while x and $\mathbf{BF}_S$ *secret* to each other.

We consider two security definitions. The one is indistinguishability of the client's input and the second one is simulatability of the malicious client. We provide a specific definition here, but the interested readers can find the generic one in [11].

Definition 1 (Client's privacy)**.** Client's privacy is defined through the following game. First the server chooses $S \subseteq U$ and $(x_0, x_1) \in U^2$. $b \in \{0, 1\}$ is randomly chosen and x_b is utilized as x according to the protocol on the client side. On the other hand, S is utilized by the server. The goal of the semi-honest server is guessing b. We denote b' for the guess. We require to the protocol that for any semi-honest server, $|\Pr[b = b'] - 1/2|$ is negligible.

Let $\mathsf{BF}(S, x, l, m)$ be the following algorithm:

Input S, x, l, m

Output $[x \in \mathbf{BF}_S]$

Algorithm Description

Step 1: Choose l hash functions $H_1, \ldots, H_l$.

Step 2: Store S in $\mathbf{BF}$ using H_i's and then verifies whether x is in the Bloom filter $\mathbf{BF}_S$.

Step 3: If this is satisfied then output 1, and 0 otherwise.

With this algorithm, the server's privacy is defined as follows: [1]

Definition 2 (Server's privacy)**.** Server's privacy is defined through the real/ideal implementation paradigm. In the ideal implementation, we assume that there is a third party T. The server sends S and the malicious client sends x' to T. Here, the input of the client is x but x' may be different from it. T computes $\mathsf{BF}(S, x', l, m)$ and returns the result to the client. Let $\mathtt{ideal}(S, x, l, m)$ be the output of the client in the ideal implementation. In the real implementation, the honest server and the malicious client proceeds the protocol without T. Let $\mathtt{real}(S, x, l, m)$ be the output of the client in the real implementation. We say that the protocol is secure if, for every x, S, l, m and the client, there is a simulator in the ideal world such that

$$\mathtt{real}(S, x, l, m) \stackrel{c}{\equiv} \mathtt{ideal}(S, x, l, m),$$

where $A \stackrel{c}{\equiv} B$ means these two random variables A and B are indistinguishable by any probabilistic polynomial-time algorithm.

[1] We only describe the situation that the client can send only one query to the server. However, extension to multiple queries is simple by applying the definition for the adaptive oblivious transfer (e.g., [14]).

3 Construction Based on Blind Signatures

3.1 Unique Blind Signature

The definition in this section is mainly extracted from [7].
A blind signature scheme BS is a tuple of algorithms $(\mathsf{Kg}, \mathsf{Sig}, \mathsf{User}, \mathsf{Vf})$.

- The signer generates a key pair (vk, sk) (a verification key and a signing key) via the key generation algorithm Kg, where vk and sk are the verification and the signing keys, respectively.
- To obtain a signature on a message m, a user and a signer engage in an interactive signing protocol dictated by $\mathsf{User}(vk, m)$ and $\mathsf{Sig}(sk)$ algorithms. At the end of the protocol, the User algorithm returns a signature or $\perp$ which indicates rejection.
- The verification algorithm $\mathsf{Vf}(vk, m, s)$ returns 1 if the signature s on a message m is deemed valid and 0 otherwise.

The correctness requires that $\mathsf{Vf}(vk, m, s) = 1$ for all (vk, sk), for all $m \in \{0,1\}^*$, and for all signatures output by $\mathsf{User}(vk, m)$ after interacting with $\mathsf{Sig}(sk)$. We say that BS is *unique* if for each verification-key $vk \in \{0,1\}^*$ and each message $m \in \{0,1\}^*$ there exists at most one signature $s \in \{0,1\}^*$ such that $\mathsf{Vk}(vk, m, s) = 1$. There are only two unique blind signature schemes appeared in the literature [5, 8]. There exist some common properties in these schemes. For example, both of them are two messages schemes (one from the user to the signer, and another from the signer to the user).

Definition 3 (One-more unforgeability)**.** This property requires to the protocol that no adversary can output $n+1$ valid message-signature pairs after given the public keys as input and after at most n interactions with a signing oracle. We say that BS is (t, q_S, ϵ)-unforgeable if there is no algorithm running in time at most t and making at most q_S signing queries has probability greater than ϵ of winning this game. We say that BS is unforgeable if, for every polynomial t and polynomial q_S, BS is (t, q_S, ϵ)-unforgeable with ϵ negligible.

By assuming the *chosen-target RSA* problem and the *chosen-target computational Diffie-Hellman* problem are hard[2], the Chaum's and Boldyreva's blind signature schemes [8, 5] satisfy above property.

Definition 4 (Blindness)**.** A signature scheme is called blind if the server's view and a message-signature pair are statistically independent.

Both schemes also satisfy this property.

3.2 Construction

In this section, we construct the privacy preserving Bloom filter protocol from the unique blind signature. In the protocol, each hash function H_i is defined by

$$H_i(x) := H(x, \mathsf{Sig}(sk, x), i).$$

Note that, in the protocol, S is stored in the Bloom filter. By these functions and by the unique blind signature, the protocol works as in Figure 2.

[2] If the client only sends *one* query to the server, then we can employ (standard) RSA and CDH problems.

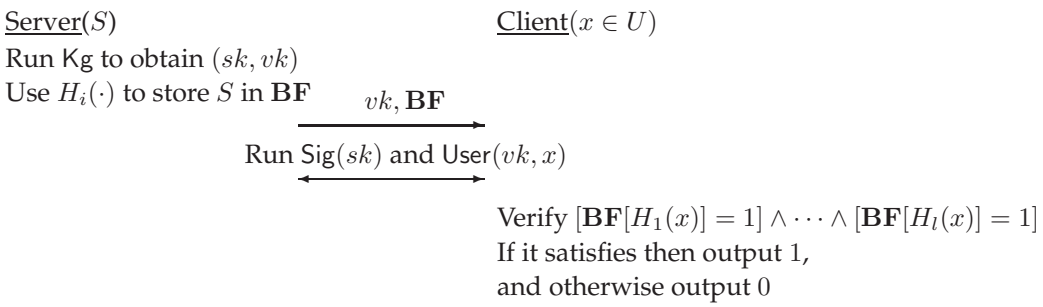

Figure 2: Privacy-preserving Bloom-filter protocol based on the blind signatures

Proposition 5. *The proposed protocol is secure against malicious clients.*

Proof. To prove the security of the proposed protocol, we need to construct simulator C^* who is the client in the ideal-implementation. Given C, $C^*(x, n, m, l)$ works as follows:

- Given (x, n, m, l), C^* generates (sk, vk) via Kg.
- C^* generates $\mathbf{BF}$ of size m. And C^* chooses nl random positions in $\mathbf{BF}$ and sets 1 to those positions.
- C^* runs C on $\mathbf{BF}, x, vk$ and runs Sig with C under a signing-key sk.
- If C^* is given the query $q := (q_1, q_2, q_3)$ with $1 \leq q_3 \leq l$ and q_2 being the signature of q_1, then C^* queries q_2 to T to obtain d.
  - If $d = 1$, C^* randomly chooses l positions $p_1, \ldots, p_l$ until $\mathbf{BF}[p_j] = 1$ for all $1 \leq j \leq l$.
  - If $d = 0$, the C^* chooses randomly chooses l positions $p_1, \ldots, p_l$ until for some $1 \leq j \leq l$ $\mathbf{BF}[p_j] = 0$.
- At the end, C^* sets $H(q_1, q_2, 1) := p_1$, ..., $H(q_1, q_2, l) := p_l$ and uses these for the input-output relation of the random oracle.
- C^* outputs whatever C outputs.

Since the blind signature scheme satisfies one-more unforgeability, there is only negligible chance to query two different message/signature pairs. Thus the simulator perfectly simulates the malicious client C. □

Proposition 6. *The proposed protocol is secure against malicious servers.*

Proof. Let us denote the view of the server by V. And let X, S be random variables of the message and the signature. Then, from the definition of the blind signature scheme, for every x_0, x_1,

$$\Pr[V \mid (X, S) = (x_0, s_0)] = \Pr[V]$$

and

$$\Pr[V \mid (X, S) = (x_1, s_1)] = \Pr[V],$$

where s_i is a signature of x_i under sk for $i = 0, 1$. Therefore, there is no algorithm which can distinguish x_0 from x_1. □

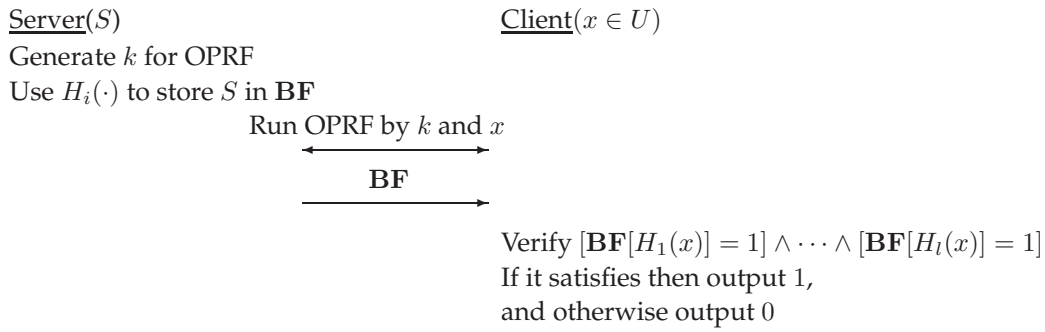

Figure 3: Privacy-preserving bloom-filter protocol based on the oblivious pseudorandom function

4 Construction Based on Oblivious Pseudorandom Functions

4.1 Oblivious Pseudorandom Function

In the previous section, we show the construction from the blind signature in the random oracle model. In this section, we show how to construct it without random oracles. The construction relies on the oblivious pseudorandom function [9]. Similar to the blind signature, there are two parties, a server and a client, who have a key k and a message x, respectively, and the goal of the protocol is the client obtaining $f_k(x)$ while keeping k and x secret to each other, where $f_k : D \to R$ is a pseudorandom function keyed by k.

We require the followings to the oblivious pseudorandom function. The one is indistinguishability of the client's input and the second one is simulatability of the malicious client.

Definition 7 (Client's privacy)**.** The client's privacy is defined through the following game. First the server chooses $x_0, x_1 \in D$. $b \in \{0, 1\}$ is randomly chosen and x_b is utilized as x according to the protocol on the client's side. On the other hand, k is utilized by the server. The goal of the semi-honest server is guessing b. We denote b' for the guess. We require to the protocol that for any semi-honest server, $|\Pr[b = b'] - 1/2|$ is negligible.

On the other hand, the following property is required for the server's privacy.

Definition 8 (Server's privacy)**.** The server's privacy is defined through the real/ideal implementation paradigm. In the ideal implementation, we assume that there is a third party T_{PRF}. The server sends k and the malicious client sends x' to T_{PRF}. Here, the input of the client is x but x' may be different from it. T_{PRF} computes $f_k(x')$ and returns the result to the client. Let $\mathtt{ideal}(k, x)$ be the output of the client in the real implementation. In the real implementation, the honest server and the malicious client proceeds the protocol without T_{PRF}. Let $\mathtt{real}(k, x)$ be the output of the client in the real implementation. We say that the protocol is secure if, for every k, x and the client, there is a simulator in the ideal world such that

$$\mathtt{real}(k, x) \stackrel{c}{\equiv} \mathtt{ideal}(k, x).$$

4.2 Construction

The construction of the privacy-preserving Bloom-filter protocol is simply replacing the blind signature and the OPRF. However, for the security proof, we change the order of the

protocol: first running OPRF and then sending the Bloom filter.

We utilize the oblivious pseudorandom function defined by the pseudorandom function

$$f : U \to R_1 \times \ldots \times R_l = [0, m-1]^l.$$

Then we define $H_j(x)$ by the value map to R_j with $f(x)$:

$$f(x) = (H_1(x), \ldots, H_l(x)).$$

The protocol is described in Figure 3.

Proposition 9. *The proposed protocol is secure against semi-honest servers.*

Proof. The requirement of the indistinguishability to the employed oblivious pseudorandom function directly contributes to the security of the proposed protocol against semi-honest servers. □

Proposition 10. *The proposed protocol is secure against malicious clients.*

Proof Sketch. This proposition can be proven in the same way as [12]. That is, we can prove this proposition in the hybrid model (see Theorem 7.4.3 in [11]). In this model, instead of running the oblivious pseudorandom function, there is a trusted party T_{PRF} who is given a key k (from the server) and a message x' (from the client). And it sends $f_k(x')$ to the client. Therefore, the simulator C^* for the malicious client C receives C's input x'. Then C^* generates $(p_1, \ldots, p_l) \in [0, m-1]^l$ randomly and sends it to C. Next C^* gives x' to the third party T to obtain $b \in \{0, 1\}$. If $b = 1$ the simulator prepares $\mathbf{BF}$ such that $\mathbf{BF}[p_j] = 1$ for every j and randomly adds one to $nl - l$ positions in $\mathbf{BF}$. If $b = 0$ the simulator prepares $\mathbf{BF}$ and randomly chooses nl positions and sets 1. If for every j $\mathbf{BF}[p_j] = 1$, then it does the same operation (until for some j $\mathbf{BF}[p_j] = 0$) again. Then C^* sends $\mathbf{BF}$ and outputs whatever C outputs.

We can show that if there is a distinguisher D which can distinguish the real-implementation from the random-implementation then f is not a pseudorandom function. □

5 Concluding Remarks

In this paper, we propose two privacy-preserving Bloom filter protocols based on the blind signature and the oblivious pseudorandom function, respectively. The comparison of these two protocols is given in Table 1. Here, the time-complexity is counted during the transfer phase between the client and the server. So we did not include the time-complexity of storing S in $\mathbf{BF}$ which is not important because the time-complexity in this phase is equivalent in both protocols. In the OPRF proposed in [9], the oblivious transfer must be invoked $\log |U|$ times and we count its time-complexity as $O(1)$ for each.

The proposed protocols in this paper are efficient theoretically. But, in some applications such as IP-traceback systems, a hash function constructed by the RSA-FDH is not suitable for those purposes due to its computational overhead. To overcome this problem, our future research direction lies in implementing the oblivious pseudorandom function based on purely symmetric primitives such as DES and AES, which will become more suitable for many applications.

Table 1: Comparison of Two Protocols

	Cryptographic Primitive	Round-Complexity	Time-Complexity
Protocol 1	Blind Signature of [8]	$O(1)$	$O(1)$
Protocol 2	OPRF of [9]	$O(1)$	$O(\log \lvert U \rvert)$

References

[1] S.M. Bellovin, W.R. Cheswick, *Privacy-Enhanced Searches Using Encrypted Bloom Filters*, Cryptology ePrint Archive, 2004/022.

[2] M. Bellare, C. Namprempre, D. Pointcheval, M. Semanko, *The Power of RSA Inversion Oracles and the Security of Chaum's RSA-Based Blind Signature Scheme*, Financial Cryptography 2001, pp.309–328, 2001.

[3] M. Bellare, P. Rogaway, *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*, ACM Conference on Computer and Communications Security 1993, pp.62–73, 1993.

[4] B.H. Bloom, *Space/time Trade-offs in Hash Coding with Allowable Errors*, Communication of the ACM 13(7), pp.422–427, 1970.

[5] A. Boldyreva, *Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme*, Public Key Cryptography 2003, pp.31–46, 2003.

[6] A. Broder, M. Mitzenmacher, *Network Applications of Bloom Filters: A Survey*, Internet Math, Volume 1, Number 4 (2003), 485-509.

[7] J. Camenisch, G. Neven, A. Shelat, *Simulatable Adaptive Oblivious Transfer*, EUROCRYPT 2007, pp.573–590, 2007.

[8] D. Chaum, *Blind Signature for Untraceable Payments*, CRYPTO 1982, pp.199–203, 1983.

[9] M.J. Freedman, Y. Ishai, B. Pinkas, O. Reingold, *Keyword Search and Oblivious Pseudorandom Functions*, TCC 2005, pp.303–324, 2005.

[10] M.J. Freedman, K. Nissim, B. Pinkas, *Efficient Private Matching and Set Intersection*, EUROCRYPT 2004, pp.1–19, 2004.

[11] O. Goldreich, *Foundations of Cryptography II: Basic Applications*, Cambridge University Press, 2004.

[12] C. Hazay, Y. Lindell, *Efficient Protocols for Set Intersection and Pattern Matching with Security Against Malicious and Covert Adversaries*, TCC 2008, pp.155–175, 2008.

[13] E.J. Goh, *Secure Indexes*, Cryptology ePrint Archive, 2003/216.

[14] W. Ogata, K. Kurosawa, *Oblivious Keyword Search*, J. Complexity 20(2-3), pp.356–371, 2004.