Efficient Private Techniques for Verifying Social Proximity

Michael J. Freedman[†*] and Antonio Nicolosi[†*]
[†]*New York University*, [*]*Stanford University*

Abstract

A variety of peer-to-peer systems use social networks to establish trust between participants. Yet the sharing of social information introduces privacy concerns. This paper proposes new privacy-preserving cryptographic protocols that enable participants to verify social proximity while exposing minimal information about the parties' social contacts. Compared to previous results, our protocols are either significantly more efficient (orders of magnitude faster than PM *[3]) or achieve stronger security properties at similar cost.*

1 Introduction

In peer-to-peer systems where resources are scarce or users are subject to abuse, participants can leverage social relationships to guide their interactions with other users. Further considering transitive trust relationships can extend a user's vantage, while still incurring a low risk of coming across abusive users. In the email or instant messaging contexts, for example, social networks can facilitate cooperative spam blacklisting [8] or sender whitelisting [4].

A naïve approach to discover transitivity is for one party to send his list of friends to the other party, who computes the set intersection of their two input sets. Yet this simple form of information sharing introduces privacy concerns.

While the problem of privacy-preserving two-party computation has been widely studied in the cryptographic literature [6, 9], general-purpose cryptographic solutions are too computationally expensive for practical use. Furthermore, their privacy guarantees are often misaligned with applications' specific threat models (discussed in §3).

This paper describes efficient cryptographic protocols with which parties can determine shared friends while exposing minimal information about their social contacts. Using RE: [4] as a motivating example—an email system we are building that reliably accepts mail from senders based on proximity in a social network—we describe two alternative methods to verify social proximity. The first method, based only on cryptographic hash functions and symmetric encryption, meets all of RE:'s current privacy and security goals at a fraction of the cost of its current Private Matching [3] protocol. The second method, while of comparable cost, achieves stronger privacy guarantees (namely, *non-transferability*) through its novel use of cryptographic properties of bilinear groups [2].

Our contributions are two-fold. First, we describe and define a security model for verifying social connectedness in a privacy-preserving fashion (§3). In fact, the mismatch between RE:'s goals and the privacy properties offered by Private Matching were a source of both computational inefficiency and privacy limitations. Second, we propose cryptographic protocols that protect such social proximity queries, for both scenarios that require high efficiency (§4.1) and those that demand strong security properties (§4.2). While this paper employs RE: to help demonstrate our protocols' use within a concrete system, they are similarly applicable to other applications that leverage social networks.

2 Motivating application: RE:

Reliable Email (RE:) [4] is an automated email acceptance system that whitelists email according to its sender. It seeks to undue the email unreliability introduced by content-based filters and other spam-fighting technologies which, while seeking to minimize the amount of spam that reaches a user's inbox, occasionally misclassify legitimate mail as spam.

The concept of sender-based whitelisting for email is hardly new. Yet, traditional whitelists suffer from two chief usability issues. First, a recipient's whitelist cannot accept mail from a sender previously unknown to the recipient. Second, populating whitelists requires manual effort distributed diffusely in time, as users acquire new contacts.

To overcome these limitations, RE: *automatically* broadens the set of senders whose mail is accepted by recipients' whitelists by explicitly examining the social network among email users. Specifically, RE: allows a user R to *attest* to another user S, which indicates that R is willing to have email from S directly forwarded to his mailbox. In other words, "User R trusts his *friend* S not to send him spam." Such an attestation is a digitally-signed statement of the form:[1]

$$\sigma_{R \to S} = \{H(R), H(S), start, end\}_{SK_R}$$

where H is a collision-resistant cryptographic hash function like SHA-256 operating on the users' email addresses, $start$ and end define the attestation's validity period, and SK_R denotes user R's signing key.

RE: leverages these attestations for accepting mail in cases where the sender S and recipient R are *not* already friends, but instead share a *bridging* friend T, resulting in a *friend-of-friend* (FoF) relationship between S and R.

By performing an FoF query, a recipient can determine which of his friends, if any, have attested to the sender. RE: achieves this while still providing the following privacy properties:

- The sender S does not learn anything about R's friends. Both learn an upper bound on the number of friends presented by the other, however.
- The recipient R learns only the intersection of the two sets of friends, *i.e.*, those T for whom R signed $\sigma_{R \to T}$ and from whom S received $\sigma_{T \to S}$.
- A third party observing all messages between S and R learns an upper bound on the size of each input, but nothing about their content nor the intersection size.
- Only R can execute the FoF query.

RE: provides the final property through its use of a one-time authorization token, while the first three properties are achieved through the use of a Private Matching (PM) protocol [3].

At a high level, PM is a two-party interactive protocol, where the input of each party is a set, and the output (learned only by one party) is the input sets' intersection. PM uses the homomorphic properties of certain public-key encryption schemes.

In RE:'s case, R's inputs are the email addresses of those X such that $\sigma_{R \to X}$, while S's inputs are those Y such that $\sigma_{Y \to S}$, along with the $\sigma_{Y \to S}$ themselves as payloads for each input. After running PM, R learns the email addresses for the set of bridging friends $\mathcal{T}$ and the corresponding attestations $\{\sigma_{T \to S} : T \in \mathcal{T}\}$. R finally verifies the digital signatures on these attestations before whitelisting S's email.

RE:'s initial concern with sharing friendship lists ("address books") for whitelisting purposes was the potential for spammers to use such a mechanism to harvest valid email addresses. RE:'s use of the PM protocol certainly prevents such an attack. It does not, however, prevent parties from "lying" about their inputs,[2] *e.g.*, by including in their input sets email addresses of people for whom they do not have the appropriate attestations.

While in this context the sender S cannot benefit from lying—as R will check the recovered attestations' signatures, match them to the supplied email addresses, and verify that the proper attestation path exists—a deceitful recipient may lie to mount a targeted attack against those parties that consider S a friend, for example. Namely, to verify whether some party Z considers S a friend, R simply claims to consider Z a friend himself when performing an FoF query with S: if S has an attestation $\sigma_{Z \to S}$, R will receive such attestation as part of PM's output.

Within RE:'s ill-defined security model, it is not even clear if and how such behavior could be construed as a protocol abuse, as R can generate an attestation $\sigma_{R \to Z}$ at any time anyway (say, with a very short duration). In the next section, we propose a more formal privacy model for verifying proximity in social networks that directly addresses these shortcomings.

3 Model

A social network can be modeled as a directed graph $G = (V, E)$, whose vertices represent the users of the system and where the presence of an arc $(R, T) \in$

[1] The original notation used in RE: [4] for attestations had the form $R \to S$; we chose to adopt a subscripted notation to reserve "plain arrows" to denote social links (see §3).

[2] Indeed, private function evaluation protocols, of which PM is an instance, are proven secure in a model that only concerns itself with preventing information leakage from the function's execution; the model does not embed a notion of "proper" inputs, so one cannot directly reason about lying parties.

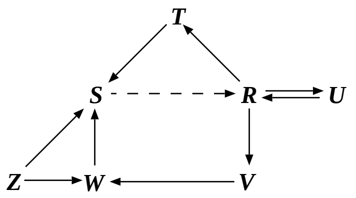

Figure 1: A fragment of a social network. Solid arrows represent trust relationships; the dotted arrow highlights a pair of users for which to verify social proximity.

E (also denoted $R \to T$) indicates the existence of a social relationship between user R and user T. We will discuss the implications attached to such relationships shortly; for now, we will just take $R \to T$ to mean that "T is R's friend."

This graph is represented within the system in a distributed fashion: each participant has only a local view of the network, consisting of its incoming and outgoing arcs. Additionally, the system provides a *proximity check* mechanism by which a user S can help R determine whether he is "close enough" to her in the social network. In particular, R can find out all *bridging* friends X such that $R \to X$ and $X \to S$. Such mechanism is exposed to a higher-level application, in which users send *requests* to each other, and requests may be treated differently by the recipient according to the social proximity of the sender (*e.g.*, whitelisting FoF's in RE:).

Figure 1 illustrates this for a fragment of a social network, where R learns that there is exactly one bridging friend between him and S, namely T. Notice that both T and W are directly connected to S, but R should not learn about W since the arc $R \to W$ does not appear in the graph.

To properly address privacy concerns of this kind, we first elaborate on the nature of the relationships represented by the social network. In the context of RE:, such relationships were viewed as predominantly unidirectional: $R \to T$ roughly corresponds to the notion that "user R trusts T not to send him spam." Under this interpretation, whether the arc $R \to T$ appears in the social network or not is essentially up to R. As we alluded in §2, however, this approach is arguably too lax: Building on the example of Figure 1, an overly curious R could unilaterally augment the social network with arcs $R \to U$, $R \to W$, $R \to Z$. This would "entitle" R to learn about the social link $W \to S$ when receiving email from S, breaching the privacy of both W and S.

To this effect, we posit that the presence of social link $R \to T$ ought to express consent of *both* parties:

(Forward Trust) User R places some form of trust on user T that T can use to demonstrate to some U the presence of a chain $U \to R$, $R \to T$.

(Backward Authorization) User T authorizes user R to discover links of the form $T \to X$ when trying to establish the existence of a social chain such as $R \to T, T \to S$.

Within a specific system, each such requirement would be associated with some concrete piece of data. For example, R's trust in T could be expressed via a digitally-signed attestation *à la* RE:, whereas backward authorization could be implemented as a shared secret key that T gives to R (as in §4).

Under such a setup, one can formalize a system's privacy properties by explicitly pointing out what information is exposed to the users, in terms of guarantees of the form: "During a proximity check with user S, user R learns at most $\mathcal{I}$." Following the approach of secure Multi-Party Computation [6, 9], a statement of this sort is proved by showing that, given $\mathcal{I}$ and the knowledge held by R (which can be deduced from R's social relationships), it is possible to simulate (or "fake") the content of all messages seen by R during the proximity check. This implies that any other information exposed to R can be derived using only $\mathcal{I}$ and R's knowledge. Thus, $\mathcal{I}$ itself provides an upper bound on the extra knowledge that R gains. We apply this proof technique in §4 to assess the privacy of our constructions.

4 Constructions

4.1 An Efficient Hash-Based Construction

Our first construction assumes, as in RE:, that each user R has a signing/verification key pair SK_R/VK_R. Additionally, R maintains a secret seed s_R for a cryptographic pseudo-random function $\mathbf{F}$ (*e.g.*, 256-bit long for HMAC-SHA-256).

Each arc in the social network is associated with a (pseudo-)random key, termed the arc's a-value. All a-values corresponding to arcs of the form $R \to X$ are derived from R's secret seed s_R as: $a_{R \to X} = \mathbf{F}_{s_R}(\text{"arc"}, R, X)$.

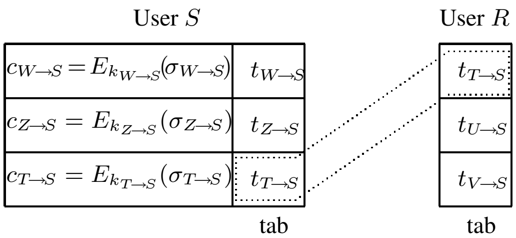

Figure 2: Data structures used for a hash-based proximity check between a sender S and recipient R.

For each social link of the form $R \to X$, user R creates an attestation $\sigma_{R\to X}$ for user X, and sends it to X along with $a_{R\to X}$ (*forward trust*). In return, R receives s_X from X (*backward authorization*).

This asymmetry in exchanging secrets stems from the way we implement proximity checks: Roughly speaking, for Y such that $Y \to S$, the sender S encrypts the attestation $\sigma_{Y\to S}$ under (a key derived from) $a_{Y\to S}$. In turn, for X such that $R \to X$, the receiver R tries to read these encrypted attestations using (a key derived from) the a-value $a_{X\to S}$ (corresponding to the possibly non-extant arc $X \to S$), which R can compute given s_X.

To help R in his decryption process (which, as described, requires a quadratic amount of symmetric-key operations), S includes a *tab* $t_{Y\to S}$ along with each encrypted attestation $c_{Y\to S}$ (*cf.* Figure 2). More in detail, for each arc $Y \to S$, S combines the attestation $\sigma_{Y\to S}$ and the a-value $a_{Y\to S}$ into a *tabbed encrypted attestation* $(c_{Y\to S}, t_{Y\to S})$ as follows. The *tab* $t_{Y\to S}$ is a pseudo-random hash computed under $\mathbf{F}$ keyed with $a_{Y\to S}$ *i.e.*, $t_{Y\to S} = \mathbf{F}_{a_{Y\to S}}(\text{“tab”}, \mathit{ReqID})$, where *ReqID* is a unique identifier supplied by the higher-level application. The ciphertext $c_Y = \mathbf{E}_{k_{Y\to S}}(\sigma_{Y\to S})$ is computed under a secure symmetric cipher $\mathbf{E}$ (*e.g.*, AES-CBC), with a key $k_{Y\to S}$ also derived from $a_{Y\to S}$: $k_{Y\to S} = \mathbf{F}_{a_{Y\to S}}(\text{“key”}, \mathit{ReqID})$.

At this point, S creates a list of these tabbed encrypted attestations, one for each of her incoming social relationships, permutes this list in random order, and sends it to R along with her request.

User R processes such a list by first looking at the tab component of each entry. In particular, for each relationship of the form $R \to X$, R holds the seed s_X. So R can form the a-value $a_{X\to S} = \mathbf{F}_{s_X}(\text{“arc”}, X, S)$, and then the $\mathbf{F}$-hash of *ReqID* (which was included as part of S's request) under $a_{X\to S}$. In this way, R computes his own set of tabs, and compares them with those received from S (which can be done efficiently, *e.g.,* by first storing one set of tabs in a hash-table, and then trying to retrieve from it the tabs of the other set). Thanks to the cryptographic properties of $\mathbf{F}$, it is extremely unlikely that two such tabs will coincide, except when they are created from the same seed. In other words, a match between the tabs guarantees that the same seed was used by both R and S, which in turn reveals the bridging friend(s), say T. At this point, R can compute the proper key $k_{T\to S} = \mathbf{F}_{a_{T\to S}}(\text{“key”}, \mathit{ReqID})$ and decrypt the corresponding encrypted attestation, thus recovering $\sigma_{T\to S}$. Finally, R verifies T's signature on $\sigma_{T\to S}$ before concluding that $R \to T$ and $T \to S$.

Security proof. Clearly, malicious senders do not pose any privacy threat, because the protocol consists just of a single sender-receiver flow. As for a malicious receiver R, we now prove that he only learns how many friends have attested to S and those attestations for which the attester is a common friend *i.e.*, $\mathcal{I} = (|\mathcal{Y}|, \{\sigma_{X\to S} : X \in \mathcal{T}\})$, where $\mathcal{Y} = \{Y : Y \to S\}$ and $\mathcal{T} = \{X : R \to X, X \to S\} \subseteq \mathcal{Y}$. To this end, we need to show how to simulate the message that R receives from S, given $|\mathcal{Y}|$, $\{\sigma_{X\to S} : X \in \mathcal{T}\}$ and the shared secrets known to R.

We start by observing that for any $W \in \mathcal{Y}\backslash\mathcal{T}$, W's random seed s_W is unknown to R, so that $a_{W\to S}$ is (pseudo-)random in R's view. Hence, by the properties of pseudo-random functions [5], it is infeasible to tell $k_{W\to S} = \mathbf{F}_{a_{W\to S}}(\text{“key”}, \mathit{ReqID})$ (resp. $t_{W\to S} = \mathbf{F}_{a_{W\to S}}(\text{“tab”}, \mathit{ReqID})$) apart from a random string $\tilde{k}_{W\to S}$ (resp. $\tilde{t}_{W\to S}$) of the same length. It follows that no efficient algorithm can distinguish $c_{W\to S} = \mathbf{E}_{k_{W\to S}}(\sigma_{W\to S})$ from $\mathbf{E}_{\tilde{k}_{W\to S}}(\sigma_{W\to S})$, which in turn, since $\mathbf{E}$ is a secure symmetric encryption scheme, cannot be distinguished from $\tilde{c}_{W\to S} = \mathbf{E}_{\tilde{k}_{W\to S}}(0^{|\sigma_{W\to S}|})$. Thus, we can replace $(c_{W\to S}, t_{W\to S})$ in S's message with a “randomized” pair $(\tilde{c}_{W\to S}, \tilde{t}_{W\to S})$, without R noticing the change.

Simulating the tabbed encrypted attestation $(c_{T\to S}, t_{T\to S})$ for $T \in \mathcal{T}$ is easier, since in this case we have $\sigma_{T\to S}$ (from $\mathcal{I}$) and $a_{T\to S}$ (as $R \to T$, and so, by backward authorization, R knows s_T, from which $a_{T\to S}$ is derived). Thus, we can directly compute $k_{T\to S} = \mathbf{F}_{a_{T\to S}}(\text{“key”}, \mathit{ReqID})$, $c_{T\to S} = \mathbf{E}_{k_{T\to S}}(\sigma_{T\to S})$ and $t_{T\to S} = \mathbf{F}_{a_{T\to S}}(\text{“tab”}, \mathit{ReqID})$.

4.2 Privacy in the Face of Collusions

Compared to the privacy properties of the PM-based protocol of RE:, our hash-based construction additionally guarantees that receivers cannot learn about attestations created by a user T without T's permission. This is in keeping with the notion of backward authorization, an aspect of our modeling missing from RE:'s original framework.

However, the kind of backward authorization implemented by the hash-based scheme is *transferable*: if user T authorizes user R to learn about attestations of the form $\sigma_{T \rightarrow X}$, R can further transfer such authorization to another user U. Then, during a proximity check with S, U would be able to discover the attestation $\sigma_{T \rightarrow S}$, even though the social link $U \rightarrow T$ is absent and so U was never back-authorized by T.

Notice that this scenario does not contradict the privacy guarantees proved in §4.1; rather, it points out the privacy implications that collusions of two or more users can have. In fact, it is unclear whether this ought to be considered a privacy problem: After all, if R and U pool their resources together, then they appear as "one and the same" to the rest of the system. Since a proximity check between S and R would have disclosed $\sigma_{T \rightarrow S}$ anyway, we may deem that U learning $\sigma_{T \rightarrow S}$ is a reasonable outcome.

In settings where user collusions are of concern, however, we may want to attain *non-transferability*: namely, only enable those users that T has *individually* authorized to actually learn about his attestations. We now describe a construction that leverages the cryptographic properties of bilinear groups to satisfy this stronger requirement.

Bilinear groups are pairs of cryptographic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of the same order q (for some large prime q), equipped with an efficiently computable map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ such that $e(g^a, h^b) = e(g, h)^{ab}$ for all $g, h \in \mathbb{G}_1$ and all $a, b \in \mathbb{Z}_q$ (*bilinearity*).[3] Typical examples of bilinear groups are based on elliptic and hyperelliptic curves (*e.g.*, [2, 7]).

Our bilinear construction exploits the bilinearity of the e map to enable users to "personalize" the secret values that they give out for backward authorization when establishing a social link (whereas in the hash-based scheme of §4.1, user R gives out the same secret s_R to all X such that $X \rightarrow R$). In particular, each user R maintains a secret exponent $s_R \in \mathbb{Z}_q$ and a public value $y_R^{\text{own}} = H_1(\text{"own"}, R)^{s_R} \in \mathbb{G}_1$, where $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ is a random oracle [1] with range in $\mathbb{G}_1$.[4] Then, R hands out $y_{R \rightarrow X}^{\text{fwd}} = H_1(\text{"fwd"}, R, X)^{s_R}$ to each X for which $R \rightarrow X$, and $y_{X \rightarrow R}^{\text{bwd}} = H_1(\text{"bwd"}, X, R)^{s_R}$ to those X for which $X \rightarrow R$. Notice that the bilinear property enables X to verify the correctness of the value received from R, since for properly computed $y_{R \rightarrow X}^{\text{fwd}}$, it must hold that $e(H_1(\text{"own"}, R), y_{R \rightarrow X}^{\text{fwd}}) \stackrel{?}{=} e(y_R^{\text{own}}, H_1(\text{"fwd"}, R, X))$, and a similar check can be performed to test the correctness of $y_{X \rightarrow R}^{\text{bwd}}$.

The proximity check protocol between S and R uses the same overall structure as that of the hash-based scheme, except that the a_* seeds for the pseudo-random function **F** are now computed as follows: For each $Y \rightarrow S$, S sets $a_{R,Y,S} = e(y_{Y \rightarrow S}^{\text{fwd}}, H_1(\text{"bwd"}, R, Y))$. Then, S can compute a tabbed encrypted attestation as before, using $a_{R,Y,S}$ in place of $a_{Y \rightarrow S}$. R computes his tabs in a similar fashion for each $R \rightarrow X$, by setting $a_{R,X,S} = e(H_1(\text{"fwd"}, X, S), y_{R \rightarrow X}^{\text{bwd}})$. The only detail to check is that, for those T such that $R \rightarrow T$ and $T \rightarrow S$, both S and R obtain the same value $a_{R,T,S}$, which readily follows by bilinearity.

Security proof. One can show that this bilinear scheme preserves the privacy of S's email contacts even in the face of collusions. The proof follows the same approach as the one used for the hash-based scheme of §4.1; we omit the details, and only point out that the hardness assumption needed for the bilinear groups is the standard *Decisional Bilinear Diffie-Hellman Assumption* [2]: Given (g, g^a, g^b, g^c) for random $g \in \mathbb{G}_1$, $a, b, c \in \mathbb{Z}_q$, it is infeasible to distinguish $e(g, g)^{abc}$ from a random value in $\mathbb{G}_2$.

5 Discussion

Multi-Hop Proximity via Memoization. Although this paper has focused on friend-of-friend relationships, our hash-based protocol also supports a weak form of detection for longer social paths. Namely, we can build a multi-hop path $R \rightsquigarrow T$ and $T \rightsquigarrow S$, whereby $Y \rightsquigarrow X$ corresponds to a path of length $\ell \geq$

[3] Technically, the map should also be *non-degenerate*: not all pairs in $\mathbb{G}_1 \times \mathbb{G}_1$ should map to the unit in $\mathbb{G}_2$.

[4] Reliance on the random oracle model is not necessary, but we decided not to pursue alternative approaches for simplicity.

1 in which Y and X have directly authorized each other (*i.e.*, X knows s_Y and Y knows $a_{Y\to X}$), yet signed attestations only exists for pairs of adjacent users on the path $Y \rightsquigarrow X$, *i.e.*, $\sigma_{Y\to I_1}, \ldots, \sigma_{I_{\ell-1}\to X}$.

To use a social path from T of length $\ell > 1$, S encrypts the entire multi-hop attestation chain within the ciphertext $c_{T\to S}$ associated to $t_{T\to S}$. Note that this protocol does not prevent observers from learning an upper bound on the length of each encrypted chain.

Privacy *vs.* Auditability. In modeling the desired privacy guarantees, one could consider a more privacy-preserving definition: users only find out whether bridging friend(s) exist, not their actual identity. However, we argue that this stronger guarantee would limit the confidence that an application can place on social proximity: although social trust is transitive to an extent, it seems imprudent to assert that transitivity will always correctly predict trust relationships between bridged parties.

In RE:'s case, for example, a user might incorrectly attest to a spammer, or he might get compromised and begin acting as a spammer. By uncovering the identity of the linking friend, our protocols provide *auditability*, which helps coping with these scenario by enabling the decision-maker to review and correct the elements that led to the wrong decision.

Non-Interactive Implementation. Unlike the PM-based approach, both methods described in §4 are non-interactive, requiring just a single message from S to R. This can significantly reduce system complexity (especially with respect to handling failures), and (for the specific case of RE:) facilitate integration with the existing e-mail infrastructure.

Symmetric Trust and Forward Security. For social networks with symmetric trust relationships (*i.e.,* where the links $Y \to X$ and $X \to Y$ are either both present in the social network, or both absent from the social network), our hash-based construction from §4.1 can be simplified by suppressing the a-values, and having the seed s_Y playing the role of $a_{Y\to X}$, for all of Y's social contacts X. Besides being conceptually simpler and computationally more efficient, this variant lends itself easily to extensions providing additional security properties, such as *forward security*, which we discuss next.

Party	Algorithm	Input sizes (number of friends) 10	100	1000
S	PM	589.7	27867.4	2490831.4
R	PM	14.7	110.9	1457.8
S	Hash	0.15	1.53	15.39
R	Hash	0.08	0.52	5.01

Table 1: Time (milliseconds) to perform privacy-preserving computations (with sender and recipient having inputs of the same sizes) for PM and hash-based protocols.

If a user S sends a request to U and no friend bridges U to S in the social network, all our constructions guarantee that U will not learn the identity of any of S's friends. Yet, as time passes and the social network evolves, a new social link may be established between U and one of S's friend (say, T). Now knowing s_T, if U has recorded S's request, he can recover T's earlier attestation to S.

Temporal correlations of this kind can be prevented in symmetric social networks by introducing time intervals in the model, and letting the s_* values evolve over time using hash chains. Namely, if the social link $T \to S$ is set up at time j_0, S gets from T the secret seed $s_T^{(j_0)}$. Then, at time j_1, S computes the tabbed encrypted attestation using the seed $s_T^{(j_1)}$ defined by the recurrence $s_T^{(i+1)} = H_s(s_T^{(i)})$, where H_s is a one-way permutation over the appropriate domain. Now, if U obtains $s_T^{(j_2)}$ from T at a later time j_2, he will not be able to use it to match the tabbed encrypted attestation that S included in her old message, because doing so would require inverting H_s.

Performance Comparison. We now compare the performance of our hash-based construction (from §4.1) to the PM protocol used in RE:[4].

We instantiated the PM protocol using its faster ElGamal variant with 1024-bit keys. The hash-based construction uses HMAC-SHA-1 and AES-CBC with 128-bit keys. In both the PM and the hash-based schemes, attestations use 1024-bit Rabin signatures. Both microbenchmarks were performed on a 2.4-GHz AMD Athlon processor (in 32-bit mode) and do not include network overhead (which are nearly identical for both). Recipients in both protocols stopped analyzing results once three bridging friends were uncovered (a configurable parameter).

Table 1 reports the performance of the two protocols. As we see, the hash-based construction is orders of magnitude faster than the public-key-based PM protocol. This table does not include the time to verify any uncovered attestations, as it is identical in both (*e.g.*, $15\mu s$ per 1024-bit Rabin signature).

6 Summary

Peer-to-peer systems may use social networks in order to establish trust between participants, yet they introduce privacy concerns when sharing such information. In this paper, we define a privacy model for verifying social proximity. We use insights from this model to propose two cryptographic protocols that protect social proximity queries: a hash-based protocol that provides similar privacy to RE:'s proposed use of PM, yet is orders of magnitude faster; and a bilinear-groups-based protocol that introduces protection against collusion.

Acknowledgments. We thank Nelly Fazio, Dahlia Malkhi, David Mazières, and Benny Pinkas for helpful discussions and comments on earlier drafts.

References

[1] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. CCS*, Nov. 1993.

[2] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO*, Aug. 2001.

[3] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, May 2004.

[4] S. Garriss, M. Kaminsky, M. J. Freedman, B. Karp, D. Mazières, and H. Yu. RE: Reliable email. In *Proc. NSDI*, May 2006.

[5] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4), Oct. 1986.

[6] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. STOC*, May 1987.

[7] A. Joux. A one round protocol for tripartite Diffie-Hellman. In *Proc. ANTS*, July 2000.

[8] J. Kong, P. O. Boykin, B. Rezaei, N. Sarshar, and V. Roychowdhury. Let your cyberalter ego share information and manage spam, May 2005. `http://arxiv.org/abs/physics/0504026`.

[9] A. Yao. Protocols for secure computations. In *Proc. FOCS*, Nov. 1982.