Towards Perfectly Secure and Deniable Communication Using an NFC-Based Key-Exchange Scheme

Daniel Bosk[1](✉), Martin Kjellqvist[2], and Sonja Buchegger[1]

[1] School of Computer Science and Communication,
KTH Royal Institute of Technology, 100 44 Stockholm, Sweden
{dbosk,buc}@kth.se

[2] Department of Information and Communication Systems,
Mid Sweden University, 851 70 Sundsvall, Sweden
martin.kjellqvist@miun.se

Abstract. In this paper we first analyse the possibility for deniability under a strong adversary, who has an Internet-wide transcript of the communication. Secondly, we present a scheme which provides the desirable properties of previous messaging schemes, but with stronger deniability under the new adversary model. Our scheme requires physical meetings for exchanges of large amounts of random key-material via near-field communication and later uses this random data to key a one-time pad for text-messaging. We prove the correctness of the protocol and, finally, we evaluate the practical feasibility of the suggested scheme.

Keywords: Deniability · Deniable encryption · Authenticated encryption · Perfect secrecy · Off-the-record · Key-exchange · Near-field communication · Surveillance

1 Introduction

We have learned a lot about modern government surveillance from the Snowden revelations starting in 2013. For our current treatment, the most interesting ones are the tapping of fibre-optic cables [13], the storage of all intercepted encrypted data [9], the search [10] and visualization capabilities [11] for all intercepted data. It is not the details that are interesting, it is the fact that one actor can collect, store and search Internet-wide transcripts of communication. This paper focuses on the possibility of deniability in this setting.

Today, GNU Privacy Guard (GPG) [18], Off-the-Record (OTR) [3] and Text-Secure [16] are among the popular services used for private communication. GPG provides standard asymmetric and symmetric encryption, intended for use with email. In 2004, Borisov, Goldberg and Brewer [3] first described the OTR due to limitations of deniability in GPG. The design goal of the protocol is to achieve strong privacy properties for users' online communication, the same properties as expected from a face-to-face conversation. The main application at the time

S. Buchegger and M. Dam (Eds.): NordSec 2015, LNCS 9417, pp. 72–87, 2015.
DOI: 10.1007/978-3-319-26502-5_6

was Instant Messaging (IM). In 2010, OpenWhisperSystems adapted the OTR messaging protocol [8] for use in the smartphone text-messaging app TextSecure.

The construction used for deniability in OTR, and the derived protocols, is based on the principle "innocent until proven otherwise". While this holds true for most civil societies, it is not true everywhere. There are circumstances in which the principle "guilty until proven otherwise" is applied instead. For these circumstances, with an adversary that can record all network traffic, it is not possible to create any false witness (proof-of-innocence) due to the deterministic nature of the protocol. In Sect. 3 we show that this allows an adversary with transcripts of all network traffic to verify any statements about the conversation using the transcripts. To thwart this we need truly deniable encryption, as defined by Canetti et al. [4], which means that we need to introduce some randomness.

1.1 Our Contributions

We start from protocols like OTR, but we assume a stronger adversary (Sect. 2). This stronger adversary model breaks some assumptions in OTR-like protocols and removes the possibility for deniability. We still want to achieve the same basic properties, e.g. mutual authentication, but we also want to have stronger deniability. In Sect. 3 we show that an adversary who can record all communication in a network can use the deterministic properties of commonly used mechanisms to reject lies about any communication.

We then outline the security properties needed and formally describe them and some results about them (Sect. 4). We continue to present a scheme which is a combination of authenticated encryption and deniable encryption (Sect. 5). This protocol is stateful, and as such, is also secure against replay and out-of-order attacks. It is a general design, so any deniable encryption and message authentication schemes with the right properties can be plugged in.

To show that this scheme is practically feasible, we present an implementation (Sect. 6). We use Near-Field Communication (NFC) in smartphones to exchange large-enough amounts of random data when two users physically meet. Later, when the users are apart, this data is used to key a One-Time Pad (OTP) for use when communicating. To estimate the feasibility of this scheme we investigate

– the order of magnitude of random data needed to be able to cover everyday text-message conversation;
– if the NFC transmission rates and the random number generation in combination with the number of physical meetings can provide high enough exchange rates in practice; and
– how the continuous key-generation in this scheme affects battery life in the device.

We answer the first question by estimating the amount of private communication for some users (Sect. 6.1). We answer the second question by estimating the required number of physical meetings for the same users (Sect. 6.2). Since we have an estimate of the amount of exchanged randomness needed and the transmission rates for the NFC protocol, we can estimate how many physical meetings and how long transfers are needed to cover the needs. For the third question, we estimate the battery usage by performing key generation and exchanges using different Android-based phones while monitoring the battery consumption (Sect. 6.3).

2 The System and Adversary Models

In our system model, we assume that we have two communication channels: one private and one public. We can implement the private channel as an NFC channel and a public network-channel, e.g. the Internet. We can always use the public channel, but we can only use the private (NFC) channel more rarely, e.g. if we are in the same physical space.

We assume that a user can, at least once, use the private channel. They must do this before any secure communication over the public channel can be started. We assume that a device can generate cryptographically strong random data and that this key-material can be securely stored on the device.

For the adversary model, we assume a stronger adversary, Eve. Eve records all traffic in the public channel. This means that she records all traffic for the entire Internet. Thus Eve has a transcript of all communication that has taken place in the public channel and any future communication will also be entered into her transcript. But Eve cannot record any communication in the private channel. Also, Eve cannot access the devices used in the communication. Instead, she will force us to reveal the keys that produced the ciphertexts in her transcript. In summary, in related works, Eve has had the role of a prosecutor who must prove things to a judge. In our model, Eve has the role of being both prosecutor and judge, which is more the case in some surveillance states.

2.1 A Formal Definition of the Adversary

More formally, we summarize Eve's capabilities in the definition below. She has the transcript of all communication in the public channel and she forces Alice to reveal the keys used for the ciphertexts in the transcribed conversation with Bob. Eve's task is to decide whether Alice is trying to lie.

Definition 1 (Deniability under Surveillance-State Attack, DEN-SS). *Let A be an efficient adversary. Let $P = ((m_i, k_i))_{i=1}^{n}$ be pairs of messages and keys and let $T = (t_i = \mathsf{Enc}_{k_i}(m_i))_{(m_i,k_i)\in P}$ be transcript of ciphertexts corresponding to the entries in P. P and T are generated by some algorithm using the encryption scheme $\mathcal{S} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$. Let ϕ be the challenge algorithm.*

First let the adversary choose (the index of) a transcript

$$i \leftarrow A(r, T),$$

where $r \overset{\$}{\leftarrow} R$ is random coins sampled from a set R. We then create two challenges, c_0 and c_1:

$$c_0 \leftarrow k_i,$$
$$c_1 \leftarrow \phi(r', i, P),$$

where $r' \overset{\not{c}}{\leftarrow} R$ *is again some random coins. Next, we choose a bit* $b \overset{\not{c}}{\leftarrow} \{0, 1\}$ *uniformly randomly. Finally, the adversary outputs a bit*

$$b' \leftarrow A(r, T, c_b).$$

We define the surveillance-state adversary's advantage as

$$\mathbf{Adv}^{DEN\text{-}SS}_{\mathcal{S},\phi}(A) = |\Pr[b = b'] - \Pr[b \neq b']| \,. \tag{1}$$

Eve's transcript T and Alice and Bob's plaintext transcript will be generated by a protocol. We we will present one in Sect. 6 for which Eve cannot win the above game. In the above, Alice uses the challenger algorithm ϕ to produce a key $k'_i \neq k_i$. The goal of our scheme is that the Eve cannot distinguish the two keys using the transcript T. Next we will outline the problems other protocols have against this adversary. We will use OTR as an example, but similar arguments can be made against similar protocols. Then we will return to our solution in Sect. 4 and onwards.

3 Why Alice and Bob Currently Must Forget Their Conversation

The security of today's popular services — GPG, OTR and TextSecure — rely on standard cryptographic mechanisms. These mechanisms provide strong security properties; in this section we outline why some of these properties are too strong for deniability in the setting where the adversary has a transcript of all communications.

GPG provides asymmetric and symmetric encryption intended to be used with email. Borisov, Goldberg and Brewer [3] have already presented arguments against GPG (and Pretty Good Privacy, PGP), but we will summarize them here. If Alice wants to send a message m to Bob, then she will encrypt it for Bob's public key k_B^{Pu}. She will then create a signature for the resultant ciphertext $c = \mathsf{Enc}_{k_B^{\mathsf{Pu}}}(m)$ with her own private key k_A^{Pr}, i.e. $s = \mathsf{Sign}_{k_A^{\mathsf{Pr}}}(H(c))$. Alice will then send the ciphertext block and the signature to Bob, and this transaction will be recorded in Eve's transcript. This scheme provides non-repudiation, i.e. Alice can not deny having sent the message m at a later time and Bob can also prove to a third party that Alice sent m. Further, Eve can also prove that Alice sent c, but she can only verify the plaintext m if Bob would reveal it to her.

In their paper, they suggested a scheme which does not have this property: the OTR messaging protocol. This protocol provides authentication for Alice and Bob, so that they can trust they are talking to the right person. But they can do no more than that, Bob can no longer prove to a third party what Alice has sent. They accomplish this by a continuous use of the Diffie-Hellman (DH) key-exchange and a Message-Authentication Code (MAC) based on symmetric keys.

We provide a simplified description here, mainly to give an understanding of the underlying ideas, see the original paper [3] for a detailed description. Alice chooses a secret exponent a and Bob chooses a secret exponent b. Alice signs g^a and sends

$$A \rightarrow B\colon g^a, \mathsf{Sign}_{k_A^{\mathsf{Pr}}}(g^a)$$

to Bob. Bob conversely sends

$$B \rightarrow A\colon g^b, \mathsf{Sign}_{k_B^{\mathsf{Pr}}}(g^b)$$

to Alice. By this time they can both compute the secret shared-key $k = g^{ab}$. Let H_E and H_M be two cryptographically secure hash functions, used for deriving encryption and MAC keys, respectively. When Alice wants to send the message m to Bob, she chooses a random a' and sends

$$A \rightarrow B\colon g^{a'}, c = \mathsf{Enc}_{H_E(k)}(i) \oplus m, \mathsf{MAC}_{H_M(k)}\left(g^{a'}, c\right),$$

where i is some counter, to Bob. Once she knows Bob has received the message she also sends the MAC key $H_M(k)$ to Bob. The next time Alice wants to send a message to Bob, she will use $k' = g^{a'b}$.

Now, Bob can no longer prove to a third party what Alice has said. This is due to the MAC being based on a secret key which Bob has access to. Also, since the encryption is done in counter mode [7], the ciphertext is malleable. This means that flipping a bit in the ciphertext, yields the same flip in the plaintext. Thus, anyone possessing the MAC key can modify the plaintext by flipping the bits in the ciphertext and then generate a new MAC.

3.1 Verifying Who Sent What

The arguments for forgeability using malleable encryption and publishing the MAC keys only hold if the adversary cannot trust the source of the transcript. This more powerful Eve (Def. 1) can ultimately trust the transcript since she collected it herself from the network. And *if* the courts trust Eve, if there are any courts, they also trust the transcript.

In this setting the forgeability property vanishes. Eve knows that no one has modified the ciphertext, she recorded in her transcript as it left Alice and arrived to Bob. She also recorded Alice publishing the MAC key used for the signature. This allows Eve to use the MAC for each ciphertext to verify them. She knows that Alice is the author of a message because she observes when Alice publishes the MAC key. Thus, Eve also knows that no one has used the malleability property, because if they did, that action would be recorded in Eve's transcript.

3.2 Verifying Encryption Keys

Furthermore, Eve also learns some information about the key from the ciphertext and MAC tag. Eve can use the MAC to discard false keys for the ciphertext.

Since Eve has $t = \mathsf{MAC}_{H_M(k)}(c)$ for a ciphertext c recorded in her transcript, she can reject a key $k' \neq k$ by verifying that $\mathsf{MAC}_{H_M(k')}(c) \neq t$. Hence, by having the MAC key depend on the encryption key, we automatically decrease the number of spurious keys and thus also reduce our possibility for deniability.

3.3 How Hard Is Deniability?

As suggested above, we have difficulty achieving deniability. This is illustrated by the following equations. Assume

$$\mathsf{Enc}_{H_E(k)}(m) = c = \mathsf{Enc}_{H_E(k')}(m')$$

and $k \neq k'$, then

$$\Pr\left[\mathsf{MAC}_{H_M(k)}(c) = \mathsf{MAC}_{H_M(k')}(c)\right] \approx \Pr\left[H_M(k) = H_M(k')\right].$$

I.e. our chance of lying about the key k, replacing it with a key k', is reduced to finding a collision for the hash function H_M. (There is also the negligible probability of $\mathsf{MAC}_x(c) = \mathsf{MAC}_{x'}(c)$ for $x \neq x'$ to consider.)

Furthermore, we find the key k' by finding the preimage of $H_E(k')$. And if the encryption system Enc is a trap-door permutation, then we will have to break that first, just to find $H_E(k')$ before we can attempt finding its preimage.

4 Required Security Properties

To be able to get deniability in our given scenario, Alice and Bob need to be able to modify the plaintext without modifying the ciphertext. They also need a MAC key independent of the encryption key. Then they can change the encryption key and the plaintext, but the ciphertext and MAC remains the same. In this section we will cover the needed security properties.

Canetti et al. gave the original formal definition of deniable encryption in their seminal paper [4]. We will give their definition of sender-deniable encryption for shared-key schemes here.

Definition 2 (Shared-key sender-deniable encryption). *A protocol π with sender S and receiver R, and with security parameter n, is a shared-key sender-deniable encryption protocol if:*

Correctness *The probability that R's output is different than S's output is negligible (as a function of n).*

Security *For any $m_1, m_2 \in M$ in the message-space M and a shared-key $k \in K$ chosen at random from the key-space K, then we have* $\Pr[\mathsf{Enc}_k(m_1) = c] \approx \Pr[\mathsf{Enc}_{k'}(m_2) = c]$.

Deniability *There exists an efficient "faking" algorithm ϕ having the following property with respect to any $m_1, m_2 \in M$. Let k, r_S, r_R be uniformly chosen shared-key and random inputs of S and R, respectively, let*

$c = \mathsf{Enc}_{k,r_S,r_R}(m_1)$ and let $(k', r'_S) = \phi(m_1, k, r_S, c, m_2)$. Then the random variables

$$(m_2, k', r'_S, c) \text{ and } (m_2, k, r_S, \mathsf{Enc}_{k,r_S,r_R}(m_2))$$

are distinguishable with negligible probability in the security parameter n.

This means that given a ciphertext $c = \mathsf{Enc}_k(m)$ and a false plaintext m', there exists a polynomial-time algorithm ϕ such that $\phi(c, m') = k'$ yields a key k' and $m' = \mathsf{Dec}_{k'}(c)$. As we illustrated in Sect. 3.3, there exists no such polynomial-time algorithm ϕ for OTR or GPG. But one encryption system for which the algorithm ϕ is trivial is the OTP.

Definition 3 (One-Time Pad). *Let $M = K = (\mathbb{Z}_2)^n$. Then let $m \in M$ be a message in the message-space M, let $k \in K$ be a uniformly chosen key in the key-space K. Then we define*

$$\mathsf{Enc}_k(m) = m \oplus k \qquad \text{and} \qquad \mathsf{Dec}_k = \mathsf{Enc}_k\,.$$

Shannon [17] proved that this scheme is perfectly secret. But this requires that the key k is as long as the message m. The key must be uniformly chosen, i.e. never reused. This is why this scheme is usually considered impractical. However, we can easily see, and it is also pointed out in [4], that the OTP fulfils Def. 2. We can simply define $\phi(m_2, c) = m_2 \oplus c$ and this would yield k' such that

$$\mathsf{Dec}_{k'}(c) = c \oplus k' = c \oplus (m_2 \oplus c) = m_2.$$

When we use an encryption scheme for communication we also want authenticity. Bellare and Namprempre [2] treats authenticated encryption and how to create composed authenticated encryption schemes. We will use the encrypt-then-MAC (EtM) composition. This means that we will encrypt and then compute a MAC tag on the ciphertext. We use the same formal definition of EtM as in [2].

Definition 4 (Encrypt-then-MAC, EtM). *Let $\mathcal{E} = (\mathsf{Keygen}^E, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme and $\mathcal{A} = (\mathsf{Keygen}^A, \mathsf{Tag}, \mathsf{Verify})$ be a message authentication scheme. We can then construct the authenticated encryption scheme $\overline{\mathcal{E}} = (\overline{\mathsf{Keygen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ as follows:*

```
function $\overline{\mathsf{Keygen}}$
    $k \xleftarrow{\$} \mathsf{Keygen}^E$
    $k^{\mathsf{MA}} \xleftarrow{\$} \mathsf{Keygen}^A$
    return $k \parallel k^{\mathsf{MA}}$
```

```
function $\overline{\mathsf{Enc}}(K, m)$
    $k \parallel k^{\mathsf{MA}} \leftarrow K$
    $c \xleftarrow{\$} \mathsf{Enc}_k(m)$
    $t \leftarrow \mathsf{Tag}_{k^{\mathsf{MA}}}(c)$
    return $c \parallel t$
```

```
function $\overline{\mathsf{Dec}}(K, C)$
    $k \parallel k^{\mathsf{MA}} \leftarrow K$
    $c \parallel t \leftarrow C$
    if $\mathsf{Verify}_{k^{\mathsf{MA}}}(c)$ then
        $m \leftarrow \mathsf{Dec}_k(c)$
        return $m$
    return $\perp$
```

OTR, for instance, uses a variant of EtM composition. It is a variant since in OTR the MAC key is derived from a master key. The results of [2] are proved for independent keys. Remember, this is one problem with the OTR that we want to avoid: the construction where the MAC key is a witness for the correct key. Instead of deriving the encryption key and the MAC key by using two different key-derivation functions on the same master key, we have to use information-theoretically independent keys.

Bellare and Namprempre [2] proved some properties about EtM: If the encryption scheme $\mathcal{E}$ provides Indistinguishability under Chosen-Plaintext Attack (IND-CPA) and the message authentication scheme $\mathcal{A}$ provides Strong Unforgeability under Chosen-Message Attack (SUF-CMA), then the EtM scheme $\overline{\mathcal{E}}$ provides Integrity of Ciphertexts (INT-CTXT) and Indistinguishability under Chosen-Ciphertext Attack (IND-CCA). Consequently, we are interested in what happens if we use a deniable encryption scheme in EtM. Since this authenticates the ciphertext, and not the plaintext, it will not interfere with our deniability. Since the key for encryption and authentication are independent, we can lie about one but not the other. We summarize this in the following theorem.

Theorem 1. *If $\mathcal{D} = (\mathsf{Keygen}^{D}, \mathsf{Enc}, \mathsf{Dec}, \phi)$ is a shared-key sender-deniable encryption scheme and $\mathcal{A} = (\mathsf{Keygen}^{A}, \mathsf{Tag}, \mathsf{Verify})$ is a message authentication scheme, then the scheme $\overline{\mathcal{D}}$ formed from the composition of $\mathcal{D}$ and $\mathcal{A}$ as in Def. 4 is also a shared-key sender-deniable encryption scheme.*

Proof. Bellare and Namprempre [2] proved that the resulting scheme $\overline{\mathcal{D}}$ inherits the security properties from the original encryption scheme $\mathcal{D}$. So the security and correctness of Def. 2 remains.

Let $K = k \| k^{\mathsf{MA}} \overset{\$}{\leftarrow} \overline{\mathsf{Keygen}}$. For any message M we have $C = c \| t = \overline{\mathsf{Enc}}_K(M)$ and $M = \overline{\mathsf{Dec}}_K(C)$. Use ϕ to derive a new k' as follows: $k' \leftarrow \phi(M, k, c, M')$, where M' is a new message such that $M \neq M'$. Now let $K' = k' \parallel k^{\mathsf{MA}}$. By the construction of $\overline{\mathsf{Dec}}$ (Def. 4) we will have $\overline{\mathsf{Dec}}_{K'}(C) = M'$. Thus the deniability property is retained as well. □

Note that the independence of the encryption key and the MAC key is crucial in the above theorem. If they are not independent, as in OTR, then this will only work if there exists an algorithm that can generate a new MAC key with the property that the MAC algorithm generates the same tag t for the same ciphertext c but with this new different key.

We will call a scheme composed as in Thm. 1 a *deniable authenticated encryption* scheme.

5 Achieving Deniability Against the Surveillance State

Due to the deniability requirements outlined above, the randomness used for encryption cannot be extended by a Pseudo-Random Number Generator (PRNG): if we do, then we are in the same situation as when we were using a trap-door permutation — we cannot efficiently find a seed to the PRNG which yields a stream that decrypts the ciphertext to the desired plaintext. Instead we generate randomness continuously and then exchange as much as needed using the private channel. This way we can use the everyday chance-encounters for exchanging the generated randomness when we meet, and then use it to key a deniable authenticated encryption scheme when physically apart.

5.1 A Protocol

Alice and Bob want to communicate securely with the possibility of deniability. They agree on using a stateful deniable authenticated encryption scheme $\overline{\mathcal{D}} = (\overline{\mathsf{Keygen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}}, \phi)$. The scheme $\overline{\mathcal{D}}$ provides Indistinguishability under Stateful Chosen-Ciphertext Attack (IND-SFCCA), Integrity of Stateful Ciphertexts (INT-SFCTXT) and shared-key sender-deniability.

Alice and Bob start by each generating a string of random bits. Alice generates the string $k_A \overset{\text{¢}}{\leftarrow} \overline{\mathsf{Keygen}}$ of length $|k_A|$. When Alice and Bob meet, Alice sends k_A over the private channel. Thus Eve cannot see this traffic. Later Alice wants to send a message to Bob over the public channel. To send the message m, Alice computes $C = c \parallel t \overset{\text{¢}}{\leftarrow} \overline{\mathsf{Enc}}_{k_A}(m)$ and sends it to Bob. We assume the scheme $\overline{\mathcal{D}}$ is stateful, so when Alice wants to send her next message to Bob she simply computes $C' \overset{\text{¢}}{\leftarrow} \overline{\mathsf{Enc}}_{k_A}(m')$ and sends C' to Bob. (We can turn any scheme into a stateful scheme by e.g. adding a counter, cf. [1].)

The protocol is unidirectional. If Bob wants to send messages to Alice, he has to do the same set up: first generate $k_B \overset{\text{¢}}{\leftarrow} \overline{\mathsf{Keygen}}$, then send the key to Alice over the private channel. After that he can encrypt messages to Alice using $\overline{\mathsf{Enc}}_{k_B}(\,\cdot\,)$. The reason we want the protocol unidirectional is to easily maintain the state of the encryption and decryption algorithms. The protocol, run once in each direction, is illustrated in Fig. 1.

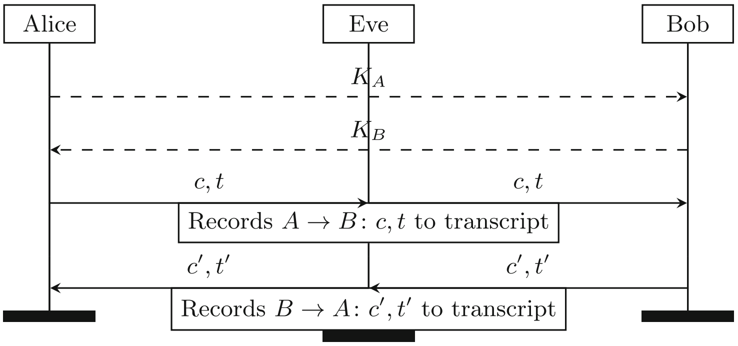
Alice
Eve
Bob
K_A
K_B
c, t
c, t
Records $A \to B$: c, t to transcript
c', t'
c', t'
Records $B \to A$: c', t' to transcript

Fig. 1. A sequence diagram illustrating the protocol. K_A and K_B are long strings of bits sent over the private channel (dashed lines), Eve cannot record this. The messaging over the public channel is full lines: $c = \mathsf{Enc}_{k_{A,m}}(m)$ and $t = \mathsf{MAC}_{k^{\mathsf{MA}}_{A,m}}(c)$; $c' = \mathsf{Enc}_{k_{B,m'}}(m')$ and $t' = \mathsf{MAC}_{k^{\mathsf{MA}}_{B,m'}}(c')$. Eve records this data as it is sent over the public channel.

5.2 The Security of the Protocol

We will now show that the protocol just described yields negligible advantage to the surveillance-state adversary (Def. 1). However, to do this we need some more details to work with the formal definition of the adversary.

Let A be an algorithm representing Alice in the above protocol description. A is a randomized algorithm which generates a sequence of messages $M = (m_i)_{i=1}^n$ of n messages. A then generates a key $k_A \xleftarrow{\$} \overline{\mathsf{Keygen}}$ by running the key generator of the scheme. Then A computes the sequence $T = (t_i \xleftarrow{\$} \overline{\mathsf{Enc}}_{k_A}(m_i))_{i=1}^n$ by encrypting each message in the sequence M. The sequence T is the transcript of the surveillance-state adversary from Def. 1. While computing T, A also stores (possibly part of) the state of the decryption algorithm $\overline{\mathsf{Dec}}_{k_A}(\,\cdot\,)$ in the following way: For each $t_i \xleftarrow{\$} \overline{\mathsf{Enc}}_{k_A}(m_i)$ operation A will extract the state $K_i = k_i \parallel k_i^{\mathsf{MA}}$ from $\overline{\mathsf{Dec}}_{k_A}(\,\cdot\,)$ so that $\overline{\mathsf{Dec}}'_{K_i}(t_i) = m_i$. We denote the sequence of states as $K = (K_i)_{i=1}^n$. And then we can form the sequence $P = M \times K$ in Def. 1 as the pairwise combination of M and K.

The first theorem shows that this scheme yields negligible advantage to the surveillance-state adversary. So our deniability properties holds.

Theorem 2 (Deniability against the surveillance state). *Given any adversary E_d against $\overline{\mathcal{D}}$, we can construct an adversary E'_d such that if E_d wins the DEN-SS game, then E'_d can distinguish between*

$$(m_2, k', r'_S, c) \text{ and } (m_2, k, r_S, \mathsf{Enc}_{k,r_S,r_R}(m_2))$$

of $\mathcal{D}$ with non-negligible probability.

Proof. Assume that E_d has non-negligible advantage in the DEN-SS game of Def. 1. Then we can construct E'_d as follows. E'_d forms $T = (c)$ and runs $i \xleftarrow{\$} E_d(T)$. Then E'_d runs $b' \leftarrow E_d(T, k)$. If $b' = 1$, then E'_d knows that k was generated using the algorithm ϕ of $\mathcal{D}$. So E'_d can distinguish (m_2, k', r'_S, c) from $(m_2, k, r_S, \mathsf{Enc}_{k,r_S,r_R}(m_2))$ with non-negligible probability. □

The next theorem states that if the deniable encryption scheme $\mathcal{D}$ provides IND-SFCCA, then so will the scheme $\overline{\mathcal{D}}$.

Theorem 3 (Chosen-ciphertext security). *Given any adversary E_p against $\overline{\mathcal{D}}$, we can construct an adversary E'_p such that*

$$\mathbf{Adv}_{\overline{\mathcal{D}}}^{\textit{IND-SFCCA}}(E_p) \leq \mathbf{Adv}_{\mathcal{D}}^{\textit{IND-SFCCA}}(E'_p) \tag{2}$$

and E'_p makes the same number of queries as E_p does.

Proof (sketch). We use a similar construction as in [2,1], i.e. construct E'_p by letting E'_p generate and add the message authentication tags itself. Then E'_p will win IND-SFCCA$_{\mathcal{D}}$ when E_p wins IND-SFCCA$_{\overline{\mathcal{D}}}$.

6 Implementation and Evaluation

We want to make a practical implementation of the protocol described above. To do this we need an encryption scheme which is deniable (Def. 2) and provides IND-SFCCA. As pointed out above, OTP provides both. We will formulate this more rigorously below. We also need a message authentication scheme providing SUF-CMA. We will use this one to achieve INT-SFCTXT. But first we need to describe the stateful use of the OTP and MACs.

Definition 5 (Stateful OTP). *Let* Keygen *be an algorithm which generates a key* k *consisting of a string of* $|k|$ *uniformly random bits. Then we define the encryption and decryption functions for a message* m *and a ciphertext* c*, respectively, as follows:*

```
function $\mathsf{Enc}(k, m)$
    State $s$ initialized to 0
    if $s + |m| > |k|$ then
        return $\perp$
    $k_m \leftarrow k[s, s + |m|]$
    $s \leftarrow s + |m|$
    $c \leftarrow m \oplus k_m$
    return $c$
```

```
function $\mathsf{Dec}(k, c)$
    State $s$ initialized to 0
    if $s + |m| > |k|$ then
        return $\perp$
    $k_c \leftarrow k[s, s + |m|]$
    $s \leftarrow s + |m|$
    $m \leftarrow c \oplus k_m$
    return $m$
```

We call the scheme $\mathcal{E} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ *a* stateful OTP *encryption scheme.*

The following theorem states that this scheme provides IND-SFCCA.

Theorem 4 (OTP implies IND-SFCCA). *If* $\mathcal{E} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ *is the stateful OTP encryption scheme (Def. 5), then* $\mathbf{Adv}_{\mathcal{E}}^{IND\text{-}SFCCA}(A)$ *is negligible for any adversary* A*.*

Proof (sketch). By construction each encryption is perfectly secure. As such the adversary's advantage is no better than random guessing.

As mentioned above, we also need to have a stateful mechanism for message authentication. For this purpose, we now describe a stateful message authentication algorithm based solely on a MAC algorithm with SUF-CMA.

Definition 6 (Stateful MACs). *Let* $\mathcal{A} = (\mathsf{Keygen}, \mathsf{Tag}, \mathsf{Verify})$ *be a message authentication scheme yielding SUF-CMA and using random bit-strings of length* $l_{\mathcal{A}}$ *as keys. Let* $\overline{\mathsf{Keygen}}$ *be an algorithm which generates a key* k *consisting of a string of* $|k|$ *uniformly random bits. Then we define the tag and verification functions for a ciphertext* c *and a tag* t*, respectively, as follows:*

```
function $\overline{\mathsf{Tag}}(k, c)$
    State $s$ initialized to 0
    if $s + l_{\mathcal{A}} > |k|$ then
        return $\perp$
    $k_c \leftarrow k[s, s + l_{\mathcal{A}}]$
    $s \leftarrow s + l_{\mathcal{A}}$
    $t \leftarrow \mathsf{Tag}_{k_c}(c)$
    return $t$
```

```
function $\overline{\mathsf{Verify}}(k, c, t)$
    State $s$ initialized to 0
    if $s + l_{\mathcal{A}} > |k|$ then
        return $\perp$
    $k_c \leftarrow k[s, s + l_{\mathcal{A}}]$
    if $\mathsf{Verify}_{k_c}(c, t) = 1$ then
        $s \leftarrow s + l_{\mathcal{A}}$
        return 1
    return 0
```

We call the scheme $\overline{\mathcal{A}} = (\overline{\mathsf{Keygen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ *a* stateful message authentication *scheme.*

Note that we only update the state if the verification is successful. We do not want to update the state in the case of an attack, that might bring the state of the verifying function out-of-sync with the state of the tagging function [1]. (Similarly for $\overline{\mathsf{Dec}}$ in Def. 4: to not bring Enc and Dec out-of-sync.)

We now show that the stateful message authentication mechanism provides INT-SFCTXT.

Theorem 5 (Stateful MACs implies INT-SFCTXT). *If the message authentication scheme $\overline{\mathcal{A}} = (\overline{\mathsf{Keygen}}, \overline{\mathsf{Tag}}, \overline{\mathsf{Verify}})$ is a stateful message authentication scheme, then $\mathbf{Adv}_{\overline{\mathcal{A}}}^{\textit{INT-SFCTXT}}(I)$ is negligible for all adversaries I.*

Proof (sketch). By construction each tag will use a new key. As such the adversary's advantage is at best to randomly guess the next key.

The private channel for the protocol is implemented using the NFC protocol with smartphones. So Alice and Bob exchange the generated keys over the NFC protocol. From a user perspective, putting two phones together "charges the deniable encryption tool". This is probably a good metaphor to build on, since it builds on the mental model of a battery. Users are already familiar with this model, and thus, when running low on randomness, fewer messages should be exchanged until another physical meeting can be arranged to "charge" the tool again.

We have developed an app[1] for Android devices which implements the above ideas. It generates randomness continuously in the background to build up a pool of randomness. It can also exchange this randomness with another phone over NFC.

6.1 The Amount of Randomness Needed

Since we use the OTP, we need as much key material for encryption as we have plaintext. We need some additional key-material for the MACs, e.g. 128–256 bits per sent message. Thus we can estimate the total amount of randomness needed by estimating the exchange rate of plaintext. To do this we analyse the Enron email dataset[2].

We are interested in personal communication, i.e. we are not interested in newsletters and the like. To filter out the newsletter category of messages, we rely on emails found in the users "sent" directory, since these are emails sent by real users.

Since this dataset contains a mix of corporate and private emails, and is fairly small, it is hard to draw any general conclusions from it. So the Enron dataset is just one example. Another dataset, communication using other media,

[1] The source code is available at URL https://github.com/MKjellqvist/OTPNFCTransfer/.

[2] The source code for the data analysis described below is available at URL https://github.com/dbosk/mailstat/. The Enron dataset is available from URL https://www.cs.cmu.edu/~./enron/enron_mail_20150507.tgz.

e.g. text messages rather than email, would probably change the observed user behaviour and these numbers. But our main goal is to get an estimate of user communication to see whether our scheme is completely infeasible or not, and we argue that this dataset lets us reach that goal.

In the Enron dataset, we found that the average message was 1000 B excluding any headers and attachments. The standard deviation was 6000 B. The large standard deviation can probably be explained by the data being emailed: If a conversation requires a few rounds, then the previous messages accumulate in the body of the email as included history.

We also found that the average user communicates with 100 other users. The standard deviation was 200. If a user sends 5 messages per day (standard deviation: 15), then we need on average less than 137 KiB per day. This means that we need less than 50 MiB to store the key-material of one year — for all users.

We use Android's "SecureRandom" to generate our randomness. This is the only supported way to generate randomness on the Android platform, and it allows us to generate enough amounts of random data. Some research [5,14] suggest that "SecureRandom" under certain circumstances uses a low entropy seed. However, the documentation states that SecureRandom can be relied upon for cryptographic purposes. With these contradictory statements, the security of SecureRandom for use with the OTP must be investigated further.

6.2 The Number of Meetings and Transfer Time

From the above analysis, we know the average amount of data communicated between users per day. We also know that the NFC protocol can achieve a transmission rate of up to 424 kbit/s [15]. Considering this, we can see that even if a user sends *ten times* the average amounts, the time required for key-exchanges is still on an order of 10s of seconds per day. (Order of minutes weekly or half-hours yearly.) This number is divided among the contacts with whom the user communicates. More frequently communicating contacts will require a larger share of the time. The times provided does not include the set up of the NFC radio channel, only actual transmission is considered. The set up phase takes about 5 s on the tested devices.

6.3 The Battery Consumption

To estimate the effects on battery consumption we find a typical RF-active rating of 60 mA for the NFC chip [15]. The battery effects of this is negligible and on the order of 2 ‰ of the battery charge at the considered usages.

To estimate the effects on battery consumption we first build a baseline. For this we used the Android systems built-in power-consumption estimates. We used one phone as a reference and two others running the app implementing our scheme.

For the component generating the randomness, tests were performed where we generated the annual demand of key-material. This provided no indication of battery drain. The processor load was measured at 2 % and the input-output load was measured at 15 %.

6.4 Some Extensions

A problem that can occur is that Alice and Bob might run out of key-material before they can meet again. One way to handle this is for them to communicate less as they are closing in on the end of their random bit strings and use the last of the randomness to schedule a new meeting.

An alternative way they can handle this problem is to switch to another scheme, but with the knowledge that it is no longer deniable. In a similar fashion, Alice and Bob might not need deniability for all their communications. Thus they can switch to e.g. OTR or TextSecure when they do not need deniability against Eve, and then switch back when they want deniability. This strategy would use less randomness and they need to meet less often.

An extension to the protocol (Sect. 5.1): Alice can do as in OTR and publish the MAC key when she receives a reply from Bob. The effect we get through this is that the MAC key is recorded in Eve's transcript, and this might lower the trust in Eve's transcript.

7 Conclusions

We set out to design a scheme which provides users with deniability in a stronger adversary model. Provided that we can generate random data with high-enough entropy, then our protocol provides perfect secrecy, authenticated and deniable encryption. However, to achieve this scheme and these properties, we require physical meetings to exchange the randomness. If Alice and Bob run out of randomness they can fall back to e.g. OTR, but then they lose deniability against Eve. In either case, they are never worse off than using OTR or TextSecure.

We also showed that our scheme is usable. We found that a typical exchange of key material requires less than 10 s daily to complete. If you exchange the key-material on a weekly basis, then it is still less than a minute, while monthly and bimonthly require up to five minutes. Thus the transmission rates are not a usability concern. Also, the effects on battery life under the considered use is not a limiting factor in neither the generation of the key-material nor the transmission of the key-material.

The method for estimating the needed amount of data can be improved. This estimate depends on the type of communication, e.g. corporate emails differs from personal text-messaging. To get more accurate estimates, it might better to evaluate a dataset from other settings. To better estimate communication needs for private individuals, it might be better to use text-messages (SMSs). However, we intended to show that our scheme is feasible, and we argue that we have reached that goal.

The only issues found in the scheme are related to the "if" regarding high-enough entropy data. The security of SecureRandom for use with the OTP must be investigated further. In addition to [14,5], we also have the result of Dodis and Spencer [6] to consider.

As a final note, the design of the NFC API is hindering the flexibility of our and similar solutions. We are mostly concerned about the following points:

– There is no mechanism in which to stream data over NFC. This is desirable from a usability standpoint of the app, in particular with regards to interrupted transmissions. This might be solved by a more innovative implementation.
– The transmission must be done in the form of files, and currently these have to reside on a publicly readable file-system on the device. This is a concern for both the confidentiality and integrity of the key-material, as the transmitted files can be intercepted by a malicious app competing for the received files.

7.1 Future Work

There are several interesting directions to follow from this work. We start with the technical one. The security of the actual NFC transfer was out of the scope of this work. However, the security of the NFC protocol must be considered: in what proximity can Eve successfully record the NFC traffic? For instance, Koscher et al. [12] found that RFID tags could be read over 50 m away in a hallway. But more interestingly, can we make usable countermeasures?

Next, we can argue the need for deniability as compared to not being able to reveal any keys. An interesting first step in this direction would be to conduct a study with users: what is the users' perception of deniability, what is more convincing? This would also be interesting to contrast by looking into game theory to see what can be said about the behaviour of a probable liar: do we gain any credibility using this deniable scheme over simply not being able to disclose the keys? What are the differences if we have a rational adversary compared to an irrational one? Finally, there is the legal perspective, which could probably also benefit from exploring these questions.

Another direction, into usable security and privacy, would be to study suitable metaphors and mental models for this kind of system. We suspect that the mental model of "charging deniability" when we exchange randomness is good, i.e. that it does not lead to any contradictory behaviours which might put the user's security and privacy at risk. Our guess is that this is more intuitive than e.g. asymmetric encryption.

Acknowledgements. This work was funded by the Swedish Foundation for Strategic Research grant SSF FFL09-0086 and the Swedish Research Council grant VR 2009-3793. We would like to thank the anonymous reviewers for valuable feedback and especially Peeter Laud for very valuable shepherding.

References

1. Bellare, M., Kohno, T., Namprempre, C.: Breaking and provably repairing the SSH authenticated encryption scheme: A case study of the Encode-then-Encrypt-and-MAC paradigm. ACM Transactions on Information and System Security (TISSEC) **7**(2), 206–241 (2004)
2. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. Journal of Cryptology **21**(4), 469–491 (2008)
3. Borisov, N., Goldberg, I., Brewer, E.: Off-the-record communication, or, why not to use PGP. In: Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society, pp. 77–84 (2004)
4. Canetti, R., Dwork, C., Naor, M., Ostrovsky, R.: Deniable encryption. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 90–104. Springer, Heidelberg (1997)
5. Ding, Y., Peng, Z., Zhou, Y., Zhang, C.: Android low entropy demystified. In: 2014 IEEE International Conference on Communications (ICC), pp. 659–664 (2014)
6. Dodis, Y., Spencer, J.: On the (non)universality of the one-time pad. In: Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science, pp. 376–385 (2002)
7. Dworkin, M.: Recommendation for Block Cipher Modes of Operation: Methods and Techniques. Special Publication 800–38A, National Institute of Standards and Technology (2001)
8. Frosch, T., Mainka, C., Bader, C., Bergsma, F., Schwenk, J., Holz, T.: How Secure is TextSecure? Cryptology ePrint Archive (2014)
9. Greenberg, A.: Leaked NSA Doc Says It Can Collect And Keep Your Encrypted Data As Long As It Takes To Crack It. Forbes (2013)
10. Greenwald, G.: XKeyscore: NSA tool collects nearly everything a user does on the internet. The Guardian (2013)
11. Greenwald, G., MacAskill, E.: Boundless Informant: the NSA's secret tool to track global surveillance data. The Guardian (2013)
12. Koscher, K., Juels, A., Brajkovic, V., Kohno, T.: EPC RFID tag security weaknesses and defenses: passport cards, enhanced drivers licenses, and beyond. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS 2009, pp. 33–42. ACM, Chicago (2009)
13. MacAskill, E., Borger, J., Hopkins, N., Davies, N., Ball, J.: GCHQ taps fibre-opticcables for secret access to world's communications. The Guardian (2013)
14. Michaelis, K., Meyer, C., Schwenk, J.: Randomly Failed! the state of randomness in current java implementations. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 129–144. Springer, Heidelberg (2013)
15. Nxp, NFC controller PN544 for mobile phones and portable equipment. 9397 750 16890. NXP Semiconductors (2010). http://www.nxp.com/documents/leaflet/75016890.pdf
16. Open Whisper Systems, TextSecure Private Messenger. https://play.google.com/store/apps/details?id=org.thoughtcrime.securesms&hl=en (accessed on November 5, 2014)
17. Shannon, C.E.: Communication theory of secrecy systems. Bell System Technical Journal **28**(4), 656–715 (1949)
18. The GnuPG Project, The GNU Privacy Guard. https://www.gnupg.org/ (accessed on June 7, 2015)