University of Toronto

Master's Thesis

Comparing Different Definitions of Secure Session

Author:
Can Zhang

Supervisor:
Charles Rackoff

January 25, 2011

1 Introduction

One of the strong goals of cryptography is to ensure secure exchange of information between two parties in the presence of an adversary who has full control over the channel. This means that the adversary decides what should be delivered to either of the parties, regardless what the two parties *wish* to deliver. In particular, we are interested in the following scenario, which we term as a "session":

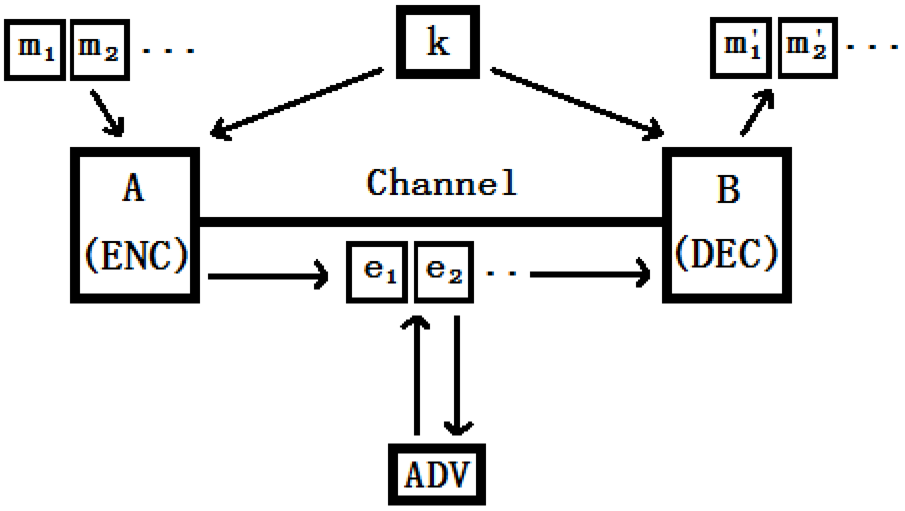

Figure 1: The Session Model

Alice (**A**) has a stream of message pieces of a fixed size $m_1, m_2, \ldots$ that she would like to share with Bob (**B**), and **A** encrypts each piece and sends the encrypted pieces to **B**. In our session model, we allow the two parties to share a random secret key for encryption and decryption. The secret key can be agreed upon in various ways: two parties can decide on a key via a trusted carrier, or they can use some public key infrastucture and a key exchange protocol where a random key can be established securely. In our discussion of sessions, we assume that the two parties start with a shared key and do not consider the distribution of the key as a part of the session. Technically, an "encrypted piece" merely refers to the text that **A** decides to put on the channel for a message piece. During the transmission of the encrypted pieces, the adversary has access to all the bits that **A** puts on the channel and decides what **B** receives. Once **B** receives (supposedly) encrypted pieces from the adversary, if the adversary did not change the encrypted pieces, **B** should be able to extract the original message pieces from the encrypted pieces using his decrypting algorithm.

For a session to be secure, two types of security properties need to be satisfied: privacy and integrity. Essentially, privacy ensures the message pieces that are embedded in the encrypted pieces remain unknown to the adversary. Integrity asserts that whenever **B** outputs a message piece, it must be the same message piece appearing at the corresponding position in the list of **A**'s input message pieces. Since the adversary decides what **B** receives, it can always send garbage to **B**. In order to detect invalid encrypted pieces, **B** has the option to output a special FAIL symbol. Whenever **B** outputs FAIL, we say that **B** "rejects" an encrypted piece. This is crucial to integrity because if **B** decrypts everything it sees, an adversary would be able to send anything to **B** and **B** would never be able to tell whether the (supposedly) encrypted pieces are unrelated to the message pieces that **A** originally has. There are notions of privacy and integrity defined in the literature for encryption primitives [5], which deal with encryption for one fixed message (in one piece), but these notions are not sufficient for session encryption. If we handle each message piece independently during a session, the adversary would be able to alter the meaning of the message pieces by simply reordering the pieces.

The first goal of this paper is to present a formal definition of "session protocol". This means that we will state the components of a session protocol and the conditions that a session protocol needs to satisfy in order to operate correctly in the absence of an adversary. Rackoff [8] proposes a simpler definition of session where **B** rejects all the subsequent (supposedly) encrypted pieces once **B** sees an invalid encrypted piece. In our definition, **B** continues normally after **B** sees an invalid encrypted piece. We define session protocol this way so that session protocols are more robust and closer to the behaviours of popular Internet protocols, which are practical implementations of sessions. With this richer definition of session protocol, we consider a number of different definitions of integrity and privacy.

The main goal of this paper is to study the notion of security of session protocols, i.e. "secure sessions". In practice, the adversary may have more power than just controlling the channel. In particular, we are interested in what information that the adversary is allowed to obtain from **B** for each (supposedly) encrypted piece, which is a form of chosen-ciphertext attack. Furthermore, in order to construct a session protocol that is most secure, we need to construct an adversary that can perform as many types of attack as possible. We will present a sequence of the definitions of integrity and a sequence of definitions of privacy, and we will show that each definition in the sequences is more powerful than the previous one.

1.1 History & Related Work

Shared-key cryptography has a long history that dates back at least to 50 BC when Julius Caesar used a cipher to send messages of military significance. Since then, various encryption techniques have been invented, but the notion of security was not rigorously discussed until late 1900s. The seminal work by Goldwasser and Micali [4] proposes definitions of security for public-key encryption primitives, and this paper had a lot of influence on future work on the definition of security. Several alternative definitions of security for shared-key encryption primitives have been proposed [1, 6], and Katz and Young [5] give a systematic analysis of those definitions and catagorize privacy and integrity (termed as indistinguishability and non-malleability by the authors) as two types of security for shared-key encryption primitives.

In constrast to substantial discussion on security for encryption primitives, there is little work on the definition of secure session, where a stream of message pieces are sent rather than a single message piece. Canetti and Krawczyk [2, 3] introduced the notion of "secure channel" as a combination of a *network authentication protocol* and a *network encryption protocol*. Both of these two protocols share a key exchange protocol, and each of them has an encryption scheme for encrypting multiple message pieces; the security of the encryption scheme depends on the security of the key exchange protocol. Namprempre [7] separates the key exchange protocol from the encryption schemes and introduces the security of secure sessions under the assumption of the sharing of a random key. In this definition of security, the definition of integrity reflects the property that message pieces are allowed to be dropped and received out of order; in our definition we insist that message pieces arrive in the same order as they are sent, and message pieces are not allowed to be dropped. On the other hand, Namprempre's definition of privacy as well as ours follows the idea of indistinguishability [5], but we differ in the constraints on the adversary in the security experiment.

1.2 The Definition of Session Protocol

A session protocol specifies how **A** encrypts message pieces and how **B** decrypts encrypted pieces in order to extract the message pieces that **A** originally has. As a basic requirement, a session protocol must ensure that the session works correctly in the absence of an adversary. The purpose of a session is to ensure secure delivery of **A**'s message pieces to **B**, and one can imagine a session protocol in which other information can be exchanged between **A** and **B** (in particular, **B** is able to send things to **A**) in order to successfully deliver the message pieces. However, we do not allow **B** to send information to **A** in our definition of session protocol, and therefore it is considered to be unidirectional in this sense. In the lower levels of the implementation of a session protocol, it might be allowed to send bits back and forth between **A** and **B**, which may be carried out insecurely, but we will ignore such lower-level interactions.

Both encryption and decryption are algorithms which operate on one message piece during each invocation, and they keep state information needed for future invocations. The reason why the protocol operates in such a stateful manner is that if session protocols are stateless, the adversary would be able to arbitrarily rearrange the message pieces without being noticed and therefore integrity could never be achieved.

In our definition of session protocol, once **B** receives an invalid encrypted piece (which **B** rejects by outputting FAIL), **B** continues to decrypt the subsequent (supposedly) encrypted pieces. This is an extension to Rackoff's original definition of session protocol [8] where **B** rejects all future (supposedly) encrypted pieces in the case of an invalid encrypted piece, which makes it easier to define the notion of security. The original definition effectively results in a more restricted adversary because the adversary does not have more opportunities to break the protocol after it sends an invalid encrypted piece to **B**. However, by allowing protocols to continue after seeing an invalid encrypted piece, sessions will not be terminated by a single invalid encrypted piece and hence become more resilient to common disruptions such as noise in the channel.

1.3 The Notion of Security of Session Protocol

As mentioned above, privacy and integrity are the two properties which session protocols must satisfy to achieve security. Technically, we need both properties to claim a session protocol secure, which means one can be sure that every message delivered by the system remains secret and unchanged. However, privacy is not always an essential property in practice if the conversation does not need to be confidential, whereas usually one will not tolerate any alteration to delivered message pieces. Therefore we construct our definition of privacy in such a way that it is most interesting and intuitively correct when the session protocol also satisfies integrity. Readers may not agree with our view on the relation between privacy and integrity, but it does not affect the notion of security, which includes integrity and privacy, because a session protocol is useful in practice only if it achieves integrity.

Our definition of integrity captures the intuitive idea that the adversary cannot trick **B** into decrypting into something different from the message pieces that **A** originally has. Recall that the adversary has complete control over the channel, and we describe an experiment which defines how successful the adversary is in breaking the integrity of a session protocol. In this experiment, a secret key is chosen for both parties and hidden from the adversary; the adversary chooses message pieces to be encrypted and sees their encrypted pieces; it also chooses (supposedly) encrypted pieces for **B** and for each such encrypted piece sees if **B** outputs FAIL. The adversary wins if **B** outputs a different message piece at some position in the stream of (supposedly) encrypted pieces from the piece at the same position in the stream of **A**'s messages pieces. Since the adversary wins as soon as **B** outputs a bad decryption, the adversary may safely assume that **B** outputs the correct message piece if it does not output a FAIL symbol. We allow the adversary to choose message pieces for **A** because it is hard to otherwise model how the messages are generated. Moreover, we allow the adversary to see **B**'s responses to the (supposedly) encrypted pieces because an attacker may be able to gain access to the local output of **B** in practice and we require session protocols to be capable of defending against such attackers as well.

The notion of privacy states that the adversary cannot distinguish the encrypted pieces of two different message pieces. As with integrity, the security experiment for privacy starts with both parties sharing a secret key. The adversary chooses message pieces for **A** and sees their encrypted pieces. For exactly once in the experiment the adversary chooses two message pieces and sees the encrypted piece of a random one of them, and the adversary needs to guess which piece of the two was encrypted. Similarly, the adversary is allowed to see for each piece whether or not **B** outputs FAIL; in a session protocol that already satisfies integrity, the adversary knows that **B** outputs the correct piece if it does not output FAIL.

Note that even if we claim a session protocol is "secure", there is a way in which an adversary can cause any session to be completely useless: whatever the adversary sees from **A**, it just sends garbage to **B** (or does not send anything at all) so that **B** does not decrypt into anything other than FAIL. This type of attack is beyond the scope of our discussion since defeating it requires a richer setting of the internet where the adversary only has control over parts of it. For the purposes of this paper, the property of integrity only ensures that all successfully decrypted message pieces must be the ones composed by **A**.

Ideally, we want session protocols to be as secure as possible, and thus we allow the adversary maximum control over the channel in our defintion. However, it is not obvious that all of the power we grant the adversary may be useful in breaking the security of a session protocol. Therefore we come up with three versions of the adversary. All three adversaries are allowed to see encrypted pieces of any message piece throughout the experiment, but the difference appears in the way in which each adversary gains information from **B**. The first and weakest adversary is only allowed to send one group of (supposedly) encrypted pieces to **B** at the end of the experiment (which means that it is not allowed to choose more message pieces for **A**) and see which ones **B** rejects; the second adversary is allowed to send any number of (supposedly) encrypted pieces to **B** one at a time, but it is still only allowed to do so after it finishes choosing message pieces for **A**; the strongest adversary is allowed to not only get information from **B** for an unlimited amount of time, but also choose message pieces for **A** and see encrypted pieces of them between rounds of (supposedly) encrypted pieces sent to **B**. Although each adversary is semantically stronger than the previous one, it is not obvious that the differences in the power of the adversary is essential in enabling it to break a session protocol. In order to show this, we will show that there are protocols which are secure under one version of the definition but insecure under a stronger one.

1.4 Pseudo-Random Function Generators

Our session protocols are constructed based on the existence of pseudo-random function generators. A function generator F maps a string $s \in \{0,1\}^n$ to a function $F_s : \{0,1\}^n \longrightarrow \{0,1\}^n$ for each integer n. Intuitively, a pseudo-random function generator produces functions that are indistinguishable from truly randomly generated functions. In order to formally talk about pseudo-randomness of a function generator F, we need to define an *adversary* for function generators. An *adversary* $D = \{D_n\}$ is a probabilistic polynomial time algorithm with an oracle for a function f as input. D outputs a bit indicating whether or not f is accepted.

We say a function generator F is pseudo-random if for every adversary D, define the following:

(1) $p_D(n) =$ the probability that if s is randomly chosen from $\{0,1\}^n$ and D is given n and an oracle for F_s, then D accepts;

(2) $r_D(n) =$ the probability that if D is given n and an oracle for a randomly chosen $f : \{0,1\}^n \longrightarrow \{0,1\}^n$, then D accepts;

Then $|p_D(n) - r_D(n)| \leq \frac{1}{n^c}$ for every c and sufficiently large n.

2 The Definition of Session Protocol

We now give our formal definition of session protocols. During a session between two parties **A** and **B**, **A** sends encrypted message pieces to **B** using *ENC*, and **B** decypts them using *DEC*, and we assert that **B** correctly receives everything that **A** sends with probability 1.

Definition. A *session protocol* consists of algorithms *ENC* and *DEC*, and length functions $z(n)$, $z'(n)$ and $z''(n)$. The length functions are computable in time polynomial in n.

ENC has the following bit strings as input: the key k of length n, a message piece m of length $z(n)$, a "history" h of length $z''(n)$, and a string of random bits *rand* if *ENC* is probablistic. *ENC* is computable in time polynomial in n, outputs a bit string of length $z'(n)$ representing the encrypted piece of m, together with a bit string of length $z''(n)$ representing the new "history" that has to be remembered in order to encrypt the subsequent pieces.

DEC has the following bit strings as input: the key k of length n, a supposedly encrypted piece e of length $z'(n)$ and a "history" h of length $z''(n)$. $DEC(k, e, h)$ is computable in time polynomial in n; it outputs either the special symbol FAIL, or of a bit string of length $z(n)$ representing the decrypted piece together with a bit string of length $z''(n)$ representing the new history that has to be remembered in order to decrypt the subsequent pieces.

In the absence of an adversary, the cryptosystem must work correctly. Specifically, the following condition must be satisfied:

Let $m_0, m_1, \ldots$ be a sequence of pieces each of length $z(n)$. Define the sequences $e_0, e_1, \ldots$ and $h_0, h_1, \ldots$ and $m'_0, m'_1, \ldots$ and $h'_0, h'_1, \ldots$ as follows, where k is an n-bit string and $rand_0, rand_1, \ldots$ are bit strings of appropriate length depending on *ENC*:

$h_0 = h'_0 =$ the all 0 string;
$(e_i, h_{i+1}) = ENC(k, m_i, h_i, rand_i)$;
$(m'_i, h'_{i+1}) = DEC(k, e_i, h'_i)$.
Then for all i, $m'_i = m_i$.

The length of each message piece $z(n)$ specifies the "granularity" of a session protocol. Since each message requires a signature for authentication, larger message sizes result in fewer pieces and hence reduces the amount of bits used for authentication. On the other hand, large message pieces take longer to encrypt and decrypt because encryption and decryption will not be able to start before the entire message piece (or encrypted piece) is received.

3 The Definitions of Integrity

In the experiment for the definition of integrity, the adversary accesses *ENC* by choosing message pieces and seeing their encrypted pieces, and it accesses *DEC* by sending (supposedly) encrypted pieces to **B** and sees whether or not **B** outputs FAIL symbols. The adversary "wins" as soon as it causes **B** to output a different message piece from the message piece at the same position in the stream of input message pieces chosen for **A**. We say that a session protocol satisfies integrity if every adversary wins with *negligible probability* (probability that diminishes as the security parameter of the session protocol grows).

Recall that we will propose three versions of definition of integrity characterized by how much and when interaction with both **A** and **B** is allowed for the adversary. In the first version, the adversary is allowed to choose message pieces for **A** and then send one batch of (supposedly) encrypted pieces to **B** at the end of the experiment; in the second version, the adversary is allowed to choose message pieces for **A** and then send (supposedly) encrypted pieces to **B** one at a time at the end of the experiment; in the last version, the adversary is allowed to choose message pieces for **A** or send (supposedly) encrypted pieces to **B** in arbitrary order at any moment during the experiment.

3.1 Definition of Integrity #1

Let $(ENC, DEC, z(n), z'(n), z''(n))$ be a session protocol. The adversary D is a polynomial size family of circuits $\{D_1, D_2, \ldots\}$, where D_n is the adversary for security parameter n. Consider the following experiment:

A random n-bit k is chosen, but D_n doesn't see it.

D_n chooses piece m_0 and sees e_0, an encrypted piece of m_0 by *ENC* using key k;

then D_n chooses piece m_1 and sees an encrypted piece e_1;

this continues for a polynomial amount of time $l = l(n)$.

Then D_n chooses a string σ and sends it to **B**.

B runs *DEC* on key k and encrypted-piece-stream σ to obtain a sequence of pieces $m'_0, m'_1, \ldots, m'_{l'}$, where each m'_i can be a string of length $z(n)$ (in which case it is a decrypted piece) or FAIL symbol.

Define $p_D(n)$ to be the probability that there exists an i, $0 \le i \le l'$ such that either of the two statements is true: (1) $i > l$; (2) $i \le l$ and $m_i \ne m'_i$ and $m'_i \ne$ FAIL.

Definition. We say a session protocol satisfies *integrity #1* if for every adversary D as described above, $p_D(n) \le \frac{1}{n^c}$ for each c and sufficiently large n.

We state that the adversary "wins" if one of the following happens: 1) the adversary causes **B** to output more pieces than what is sent; 2) the adversary causes **B** to output a different piece from the piece which is in the same position in the sequence of messages which **A** encrypts. In fact, the definition would not become weaker if we insisted that the adversary can never cause **B** to output more pieces than what is sent. Suppose that there exists an adversary D that causes **B** to output extra pieces — after outputting pieces $m'_1, m'_2, \ldots, m'_l$, **B** continues to output pieces $m'_{l+1}, \ldots, m'_{l'}$. Then we can build a new adversary D' who chooses $m_{i+1}, \ldots, m_{l'}$ arbitrarily (as long as they are not equal to $m'_{l+1}, \ldots, m'_{l'}$ respectively) after choosing the first l message pieces, and now the "extra" message pieces $m'_{l+1}, \ldots, m'_{l'}$ are "incorrect" message pieces.

3.2 Definition of Integrity #2

Let $(ENC, DEC, z(n), z'(n), z''(n))$ be a session protocol. The adversary D is a polynomial size family of circuits $\{D_1, D_2, \ldots\}$, where D_n is the adversary for security parameter n. Consider the following experiment:

A random n-bit k is chosen, but D_n doesn't see it.

D_n chooses piece m_0 and sees e_0, an encrypted piece of m_0 by ENC using key k;

then D_n chooses piece m_1 and sees an encrypted piece e_1;

this continues for a polynomial amount of time $l = l(n)$.

Then D_n chooses a string e'_0 (of length $z'(n)$), sends it to **B**; **B** runs DEC on key k and e'_0 to obtain its decryption m'_0 (a string of length $z(n)$ or FAIL symbol); then **B** lets D_n know whether the decryption is a FAIL symbol;

then D_n chooses e'_1, sends it to **B**; **B** runs DEC on key k and e'_1 to obtain its decryption m'_1 (a string of length $z(n)$ or FAIL symbol); then **B** lets D_n know whether the decryption is a FAIL symbol;

this continues for a polynomial amount of time $l'(n)$.

Define $p_D(n)$ to be the probability that there exists an i, $0 \leq i \leq l'$ such that either of the two statements is true: (1) $i > l$; (2) $i \leq l$ and $m_i \neq m'_i$ and $m'_i \neq$ FAIL.

Definition. We say a session protocol satisfies *integrity #2* if for every adversary D as described above, $p_D(n) \leq \frac{1}{n^c}$ for each c and sufficiently large n.

This definition is the same as the previous version except that the adversary now can view the FAIL/"NOT FAIL" results which **B** outputs before deciding what to send to **B** next. Intuitively the responses from **B** might provide information related to the secret key and hence help the adversary break the system. While this is true under our definition of session protocol, if we let the protocol terminate once **B** outputs a FAIL symbol as in the original definition of session protocol [8], then the responses from **B** would not help the adversary at all: the adversary should assume that the i-th piece is decrypted correctly because the adversary has already won if it is decrypted incorrectly, and the adversary has no more chances of breaking the system if **B** outputs FAIL.

3.3 Definition of Integrity #3

Let $(ENC, DEC, z(n), z'(n), z''(n))$ be a session protocol. The adversary D is a polynomial size family of circuits $\{D_1, D_2, \ldots\}$, where D_n is the adversary for security parameter n. Consider the following experiment:

A random n-bit k is chosen, but D_n doesn't see it.

D_n chooses to perform one of the following actions:

(1) D_n chooses a message piece and sees its encrypted piece;

(2) D_n chooses a string of length $z'(n)$, sends it to **B**; **B** runs DEC on key k and the string to obtain its decryption (a string of length $z(n)$ or FAIL symbol); then **B** lets D_n know whether the decryption is a FAIL symbol.

then D_n again chooses to perform one of the actions;

this continues for a polynomial amount of time. We name the message stream and encrypted-piece stream in (1) $m_0, m_1, \ldots$ and $e_0, e_1, \ldots$, respectively; also, we name the string pieces and their decryptions in (2) $e'_0, e'_1, \ldots$ and $m'_0, m'_1, \ldots$, respectively.

Define $p_D(n)$ to be the probability that there exists an i, such that $m_i \neq m'_i$ and $m'_i \neq$ FAIL.

Definition. We say a session protocol satisfies *integrity #3* if for every adversary D as described above, $p_D(n) \leq \frac{1}{n^c}$ for each c and sufficiently large n.

This version of integrity allows maximum interaction with **A** and **B** for the adversary since now there is no restriction on how and when the adversary could access *ENC* and *DEC*. In particular, the adversary may choose more message pieces after seeing some responses from **B**. It is worth pointing out that in this experiment, the adversary can ask for the decryption of message pieces even before it chooses any message piece for **A**. However, an adversary should never be able to construct the encrypted piece of a certain message piece without seeing its encrypted piece (i.e. "forging a message piece"), and therefore a secure session should be able to defend against this kind of attack.

3.4 Relations Between Different Definitions of Integrity

We will show that all three versions of the definition of integrity are different from each other. Specifically, we will show that the the Definition of Integrity #2 is stronger than the Definition of Integrity #1, and the Definition of Integrity #3 is stronger than the Definition of Integrity #2. Recall that each of our session protocols is constructed based on a pseudo-random function generator. In the proof of each theorem, we construct a session protocol, and show: 1) if there exists an adversary that breaks the integrity of this session protocol under the weaker definition of integrity, then there is an adversary which breaks the pseudo-randomness of the underlying function generator; 2) there is an adversary which breaks the integrity of this session protocol under the stronger definition of integrity. In the proof of 1), an adversary that breaks the "pseudo-randomness" of a function generator is essentially an adversary that distinguishes functions that are generated by pseudo-random function generators from functions that are generated by truly random ones, and it outputs a bit about whether or not the function is generated by a pseudo-random function generator.

Theorem 3.1. *There exists a session protocol that satisfies the Definition of Integrity* #1 *but does not satisfy the Definition of Integrity* #2.

Proof. To show that the responses from **B** may be useful to the adversary, we construct a session protocol in which *DEC* reveals one bit of the secret key each time *DEC* is called using encrypted pieces of a special format, and therefore the adversary can learn the key and encrypt anything of its choice. Essentially, the adversary only needs one additional round of access to *DEC* to send a different message piece encrypted with the key after sending a batch of (supposedly) encrypted pieces to **B** (as in the Definition of Integrity #1) in order to break this session protocol under the Definition of Integrity #2.

Consider the following session protocol $\Pi = (ENC, DEC)$:

Let F be a pseudo-random function generator where $F_i : \{0,1\}^{2n} \longrightarrow \{0,1\}^n$ for any integer i.

For an n-bit key $k = k_0 k_1 \ldots k_{n-1}$ and $z(n)$-bit message pieces $m_0, m_1, \ldots$, *ENC* encrypts each m_i by $e_i = [m_i, F_k(\bar{\imath} m_i), 0]$.

For a sequence of strings $e'_0, e'_1, \ldots$, each of length $2n+1$, *DEC* decrypts each e'_i as follows:

Let $e'_i = [m'_i, \beta'_i, b]$, where **B** is a single bit and controls the "mode" of *DEC*. When $b = 1$, *DEC* indicates if the i-th bit of the secret key k is 0 or 1 by outputting FAIL or not. When $b = 0$, *DEC* decrypts normally. More formally:

(1) If $i < n$, $b = 1$ and $k_i = 0$, output FAIL.

(2) Otherwise, check if $\beta'_i = F_k(\bar{i}m_i)$. If so, output m'_i; if not, output FAIL.

Claim 3.1.1. *Session protocol* $\Pi = (ENC, DEC)$ *satisfies the Definition of Integrity* #1.

Proof. Suppose that an adversary C breaks the integrity of Π under the Definition of Integrity #1. We will construct an adversary D that breaks the pseudo-randomness of the underlying function generator F. Let p_C be the probability defined in the Definition of Integrity #1, and assume that there exists a constant c such that for infinitely many n, $p_C(n) > \frac{1}{n^c}$; fix such an n.

Suppose we are given a function $f : \{0,1\}^{2n} \longrightarrow \{0,1\}^n$. D_n simulates C_n as follows:

Whenever C_n chooses m_i, we see $f(\bar{i}m_i)$ and give $[m_i, f(\bar{i}m_i), 0]$ to C_n. Suppose this continues for l times. Then C_n outputs a sequence of strings $[m'_0, \beta'_0, b_0], [m'_1, \beta'_1, b_1], \ldots, [m'_{l'}, \beta'_{l'}, b_{l'}]$.

D_n will accept if there is an i, $0 \le i \le l'$, such that $\beta'_i = f(\bar{i}m'_i)$, and either of the two statements is true: (1) $i > l$; (2) $i \le l$ and $m_i \ne m'_i$ and $m'_i \ne$ FAIL.

Therefore a pseudo-randomly generated f will be accepted with probability $> \frac{1}{n^c}$.

If f is randomly generated, it will only be accepted if C_n guesses the value of f on an input it did not query. Specifically, C_n has l' guesses, and therefore its probability of success is
$1 - (1 - \frac{1}{2^n})^{l'} < l' \cdot \frac{1}{2^n}$.

□

Claim 3.1.2. *Session protocol* $\Pi = (ENC, DEC)$ *does not satisfy the Definition of Integrity* #2, *i.e. there exists an adversary that breaks the integrity of* Π *under the Definition of Integrity* #2.

Proof. We will show how to construct such an adversary D. Specifically, we will construct D_n for any n.

D_n chooses arbitrary message pieces $m_0, m_1, \ldots, m_n, m_{n+1}$ and sees their respective encrypted pieces $e_0, e_1, \ldots, e_n, e_{n+1}$. Let $k = k_0 k_1 \ldots k_{n-1}$.

For each $e_i = [m_i, \beta_i, 0]$, let $e'_i = [m_i, \beta_i, 1]$. (That is, set the last bit of e_i to 1). D_n sends $e'_0, e'_1, \ldots, e'_n$ to **B** one at a time; if **B** outputs FAIL on e'_i, the adversary knows that $k_i = 0$; otherwise $k_i = 1$. Therefore D_n learns k by repeating this n times.

With key k, the adversary simply encrypts any different message other than m_{n+1} and sends its encrypted piece to **B** to make **B** output something different.

□

We simply combine both claims to conclude the proof of the theorem.

□

Theorem 3.2. *There exists a session protocol that satisfies the Definition of Integrity* #2 *but does not satisfy the Definition of Integrity* #3.

Proof. We will construct a session protocol where *ENC* reveals the secret key when asked for the encrypted piece of a specific message piece, and this message piece is revealed the same way in which the secret key is in the session protocol for the previous theorem. Therefore the adversary needs to access *ENC* at least once after sending (supposedly) encrypted pieces to **B** in order to reveal the secret key and hence break the protocol.

Consider the following session protocol $\Pi = (ENC, DEC)$:

Let F be a pseudo-random function generator where $F_i : \{0,1\}^{2n} \longrightarrow \{0,1\}^n$ for any integer i.

For an n-bit key k and $z(n)$-bit message pieces $m_0, m_1, \ldots$, *ENC* sees if any message piece is the "special" message piece predetermined by the protocol; if so, *ENC* reveals the secret key in its output, and encrypts normally otherwise. Specifically, *ENC* encrypts each m_i as follows:

(1) $e_i = [m_i, F_k(\overline{i+1} \cdot m_i), 0, \bar{0}]$, if $m_i \neq F_k(\bar{0} \cdot \bar{0})$;

(2) $e_i = [m_i, F_k(\overline{i+1} \cdot m_i), 0, \bar{k}]$, if $m_i = F_k(\bar{0} \cdot \bar{0})$.

For a sequence of strings $e'_0, e'_1, \ldots$, each of length $3n+1$, *DEC* decrypts each e'_i as follows:

Let $e'_i = [m'_i, \beta'_i, b_i, s_i]$ and $F_k(\bar{0} \cdot \bar{0}) = m^* = m^*_0 m^*_1 \ldots$. b_i controls the "mode" of *DEC* and when b_i is 1, it reveals the i-th bit of m^*, which is the "special" message piece.

(1) If $i < n$, $b_i = 1$ and $m^*_i = 0$, output FAIL;

(2) otherwise, check if $F_k(\overline{i+1} \cdot m'_i) = \beta'_i$. If so, output m'_i; otherwise output FAIL.

Claim 3.2.1. *Session protocol $\Pi = (ENC, DEC)$ satisfies the Definition of Integrity #2.*

Proof. Suppose that an adversary C breaks the integrity of Π under the Definition of Integrity #1. We will construct an adversary D that breaks the pseudo-randomness of the underlying function generator F. Let p_C be the probability defined in the Definition of Integrity #2, and assume that there exists a constant c such that for infinitely many n, $p_C(n) > \frac{1}{n^c}$; fix such an n.

Suppose we are given a function $f : \{0,1\}^{2n} \longrightarrow \{0,1\}^n$. D_n simulates C_n as follows:

Whenever C_n chooses m_i, D_n simulates *ENC* as follows:

(1) D_n checks if $m_i = f(\bar{0} \cdot \bar{0})$; if so, it accepts f.

(2) Otherwise, D_n gives $[m_i, f(\overline{i+1} m_i), 0, \bar{0}]$ to C_n.

Whenever C_n chooses to see the decryption of a string $e'_i = [m'_i, \beta'_i, b_i, s_i]$, D_n simulates *DEC* as follows:

(1) D_n checks if $b_i = 1$. If so, it checks if the i^{th} bit of $f(\bar{0} \cdot \bar{0})$ is 0; it gives FAIL to C_n if so.

(2) In cases where either of the two checks in (1) fails, D_n checks if $\beta'_i = f(\overline{i+1} m'_i)$. If so, it gives "NOT FAIL" to C_n; it gives FAIL to C_n otherwise.

(3) At any point during the experiment, let l be the number of message pieces C_n has chosen and l' be the number of (supposedly) encrypted pieces C_n has chosen to see. D_n will accept if there is an i, $0 \leq i \leq l'$, such that either one of the following statments is true: 1) $i > l$; 2) $i \leq l$, $\beta'_i = f(\overline{i+1} m'_i)$, $m_i \neq m'_i$ and $m'_i \neq$ FAIL.

Therefore a pseudo-randomly generated f will be accepted in two cases: 1) C_n chooses $F(\bar{0}\cdot\bar{0})$; 2) C_n never chooses $F(\bar{0}\cdot\bar{0})$ and tries to break the system normally. f is accepted with probability 1 in 1), and is accepted with probability $> \frac{1}{n^c}$ in 2) because D_n runs a perfect simulation of C_n. Therefore C_n accepts a pseudo-randomly generated f with a probability higher than $\frac{1}{n^c}$.

If f is randomly generated, it will be accepted if one of the two things happens: 1) C_n chooses $F(\bar{0}\cdot\bar{0})$; 2) C_n guesses the value of f on an input it did not query. Case 1) happens with probability $< \frac{l}{2^n}$ and case 2) happens with probability $< \frac{l'}{2^n}$. Therefore, the overall probability that a randomly generated f is accepted is $< \frac{l+l'}{2^n}$ which is a negligible function of n.

$\square$

Claim 3.2.2. *Session protocol* $\Pi = (ENC, DEC)$ *does not satisfy the Definition of Integrity* $\#3$*, i.e. there exists an adversary that breaks the integrity of* Π *under the Definition of Integrity* $\#3$.

Proof. We will show how to construct such an adversary D. Specifically, we will construct D_n for any n.

D_n chooses arbitrary message pieces $m_0, m_1, \ldots, m_n, m_{n+1}$ and sees their respective encrypted pieces $e_0, e_1, \ldots, e_n, e_{n+1}$. Let $F_k(\bar{0}\cdot\bar{0}) = m^* = m_0^* m_1^* \ldots m_n^*$.

For each $e_i = [m_i, \beta_i, 0, \bar{0}]$, let $e_i' = [m_i, \beta_i, 1, \bar{0}]$. D_n sends $e_0', e_1', \ldots, e_n'$ to **B** one at a time; if **B** outputs FAIL on e_i', the adversary knows that $m_i^* = 0$; otherwise $m_i^* = 1$. Therefore D_n learns $F_k(\bar{0}\cdot\bar{0})$ by repeating this n times.

Then D_n sends $F_k(\bar{0}\cdot\bar{0})$ to ENC and learns k as the last tuple of the encrypted piece. With key k, the adversary simply encrypts any different message other than m_{n+1} and sends its encrypted piece to **B** to make **B** output something different.

$\square$

We simply combine both claims to conclude the proof of the theorem.

$\square$

4 The Definitions of Privacy

In the experiment for the definition of privacy, the adversary first chooses a piece index for which it will later choose two message pieces and see the encrypted piece of a random one of them. Then the adversary accesses ENC by choosing message pieces and seeing their encrypted pieces, and it accesses DEC by sending (supposedly) encrypted pieces to **B** and sees if **B** outputs FAIL. Finally, the adversary outputs a guess at which one of the two pieces was encrypted. The adversary "wins" if its guess is correct. We say that a session protocol satisfies privacy if every adversary wins with $\frac{1}{2}$ + negligible probability.

Similar to integrity, we propose three versions of the definition of privacy characterized by how much and when interaction with both **A** and **B** is allowed for the adversary, and the differences between these versions are analogous to the definitions of integrity.

4.1 Definition of Privacy $\#1$

Let $(ENC, DEC, z(n), z'(n), z''(n))$ be a session protocol. The adversary D is a polynomial size family of circuits $\{D_1, D_2, \ldots\}$, where D_n is the adversary for security parameter n. Consider the following experiment:

D_n chooses an integer r; D_n will be trying to learn what the r-th piece of the message is.

A random n-bit k is chosen, but D_n doesn't see it.

D_n chooses piece m_0 and sees e_0, an encrypted piece of m_0 by *ENC* using key k;

then D_n chooses piece m_1 and sees an encrypted piece e_1;

this continues through piece m_{r-1} and e_{r-1}.

Then D_n chooses two pieces m^0 and m^1.

A random bit **B** is chosen but D_n doesn't see it, and then D_n sees an encrypted piece e_r of m^{ch}. We call m^{ch} the "challenge encryped piece".

Then D_n chooses piece m_{r+1} and sees an encrypted piece e_{r+1};

then D_n chooses piece m_{r+2} and sees an encrypted piece e_{r+2};

this continues for a polynomial amount of time.

Then D_n outputs a string σ and sends it to **B**. **B** runs *DEC* on key k and each piece in the encrypted-piece-stream σ, and then D_n learns if and for which pieces **B** outputs FAIL.

Then D_n outputs ch', a guess at ch.

Define $q_D(n) = \textbf{probability}(ch = ch')$.

Definition. We say a session protocol satisfies *privacy #1* if for every adversary $D = \{D_n\}$ as described above, $q_D(n) \leq \frac{1}{2} + \frac{1}{n^c}$ for each c and sufficiently large n.

We ask the adversary to choose the piece on which it will make a guess rather than choose after seeing some (supposedly) encrypted pieces (and decryptions for stronger definitions described below). It seems that we provide less information to the adversary, but in fact the definition would not be weakened if we let the adversary choose the piece whenever it wishes. Suppose there exists an adversary D which breaks a session protocol by choosing the piece to guess after interacting with **A** and **B**, we can construct an adversary D' that breaks the same protocol by choosing the piece beforehand: D' simply randomly picks a message piece to guess and imitates D until D decides on which piece it is going to guess. If D happens to choose the one on which D' randomly "decides", D' keeps imitating D and will perform equally well as D; otherwise, D' flips a coin to make a guess. Suppose D chooses at most m message pieces, and then the advantage which D' has over a half is reduced by a factor of m, which is still non-negligible.

If we let the adversary see the decryption of the challenge encrypted piece, the adversary would be able to trivially break any protocol. However, the adversary obtains the same amount of information from a "FAIL/NOT FAIL" response as it from the actual decryption for any other (supposedly) encrypted pieces. As mentioned earlier, our notion of privacy is most interesting and intuitive when the session protocol also achieves integrity. Therefore if the session protocol satisfies integrity, the adversary could safely assume that B decrypts correctly when B responds with "NOT FAIL".

4.2 Definition of Privacy #2

Let $(ENC, DEC, z(n), z'(n), z''(n))$ be a session protocol. The adversary D is a polynomial size family of circuits $\{D_1, D_2, \ldots\}$, where D_n is the adversary for security parameter n. Consider the following experiment:

D_n chooses an integer r; D_n will be trying to learn what the r-th piece of the message is.

A random n-bit k is chosen, but D_n doesn't see it.

D_n chooses piece m_0 and sees e_0, an encrypted piece of m_0 by *ENC* using key k;

then D_n chooses piece m_1 and sees an encrypted piece e_1;

this continues through piece m_{r-1} and e_{r-1}.

Then D_n chooses two pieces m^0 and m^1.

A random bit ch is chosen but D_n doesn't see it, and then D_n sees an encrypted piece e_r of m^{ch}.

Then D_n chooses piece m_{r+1} and sees an encrypted piece e_{r+1};
then D_n chooses piece m_{r+2} and sees an encrypted piece e_{r+2};
this continues for a polynomial amount of time.

Then D_n chooses a string e'_0 (of length $z'(n)$), sends it to **B**, and sees if **B** outputs FAIL.
then D_n chooses e'_1, sends it to **B**, and sees if **B** outputs FAIL;
this continues for a polynomial amount of time.

Then D_n outputs ch', a guess at **B**.

Define $q_D(n) = \textbf{probability}(ch = ch')$.

Definition. We say a shared-private-key encryption scheme satisfies *privacy #2* if for every adversary $D = \{D_n\}$ as described above, $q_D(n) \le \frac{1}{2} + \frac{1}{n^c}$ for each c and sufficiently large n.

Similarly to integrity, Privacy #2 allows the adversary to send (supposedly) encrypted pieces to B one at a time so that it may choose subsequent (supposedly) encrypted pieces according to **B**'s responses.

4.3 Definition of Privacy #3

Let $(ENC, DEC, z(n), z'(n), z''(n))$ be a session protocol. The adversary D is a polynomial size family of circuits $\{D_1, D_2, \ldots\}$, where D_n is the adversary for security parameter n. Consider the following experiment:

D_n chooses an integer r; D_n will be trying to learn what the r-th piece of the message is.

A random n-bit k is chosen, but D_n doesn't see it.

D_n chooses to perform one of the following actions:

(1) D_n chooses a message piece and sees its encrypted piece. If this is the r-th time D_n has chosen to perform this action, D_n chooses two pieces m^0 and m^1 instead; a random bit ch is chosen but D_n doesn't see it, and then D_n sees an encrypted piece e_r of m^{ch}.

(2) D_n chooses a string of length $z'(n)$, sends it to **B**, and sees if **B** outputs FAIL;

then D_n again chooses to perform one of the actions;
this continues for a polynomial amount of time. We name the message stream and encrypted-piece stream in (1) $m_0, m_1, \ldots$ and $e_0, e_1, \ldots$, respectively; also, we name the string pieces in (2) $e'_0, e'_1, \ldots$.

Then D_n outputs ch', a guess at ch.

Define $q_D(n) = \textbf{probability}(ch = ch')$.

Definition. We say a session protocol satisfies *privacy #3* if for every adversary $D = \{D_n\}$ as described above, $q_D(n) \le \frac{1}{2} + \frac{1}{n^c}$ for each c and sufficiently large n.

Similarly to integrity, this is the case where the adversary can access DEC before and while choosing message pieces for **A**.

4.4 Relations Between Different Definitions of Privacy

Since our definition of privacy is most interesting where session protocols already satisfy integrity, it makes little sense to discuss protocols which satisfy privacy but not integrity. We adopt the strongest notion of integrity, i.e. the Definition of Integrity #3. In general, integrity can be achieved by attaching a "signature" to each message piece, and therefore we can achieve both privacy and integrity by attaching signatures to encrypted pieces that satisfy privacy.

Similarly to integrity, we will show that the Definition of Privacy #2 is stronger than the Definition of Privacy #1, and the Definition of Privacy #3 is stronger than the Definition of Privacy #2. In the proof of each theorem, we construct a session protocol and show: 1) if there exists an adversary that breaks the privacy of this session protocol under the weaker definition of privacy, then there exists an adversary that also breaks the pseudo-randomness of one of the underlying function generators; 2) if there exists an adversary that breaks the integrity of this session protocol under the Definition of Integrity #3, then there exists an adversary that breaks the pseudo-randomness of the other one of the underlying function generators; 3) there is an adversary that breaks the integrity of this session protocol under the stronger definition of integrity.

Theorem 4.1. *There exists a session protocol that satisfies the Definition of Privacy* #1 *and the Definition of Integrity* #3*, but does not satisfy the Definition of Privacy* #2*.*

Proof. As in the session protocol Theorem 3.2, we construct a session protocol that reveals one bit of a "password" in each output of DEC, and by later providing the "password" to DEC it reveals whether $\bar{0}$ or $\bar{1}$ was encrypted into the challenge encrypted piece.

Consider the following session protocol $\Pi = (ENC, DEC)$:

Let F, G be pseudo-random function generators, where $F_k : \{0,1\}^n \longrightarrow \{0,1\}^n$, and $G_{k'} : \{0,1\}^{2n} \longrightarrow \{0,1\}^n$ for any n-bit string k, k'. Let kk' be the session key, where k and k' are the n-bit keys for F and G, respectively.

For $z(n)$-bit message pieces $m_0, m_1, \ldots$, ENC encrypts each m_i by $e_i = [\alpha_i, \bar{0}, 00, G_{k'}(\bar{i}\alpha_i)]$, where $\alpha_i = m_i \oplus F_k(\overline{i+1})$. Here α_i is an encrypted piece that satisfies privacy #1, and we sign α_i to satisfy integrity. Notice that we are not signing the two middle fields because it is crucial to be able to change those bits in order to trigger the special modes where DEC reveals critical information. Also, this still satisfies integrity since those bits are indepedent of the content of the message pieces and changing them will not result in a different message when decrypted.

For a sequence of strings $e'_0, e'_1, \ldots$, each of length $3n+1$, DEC decrypts each e'_i as follows:

Let $e'_i = [\alpha'_i, pwd_i, b_i, \beta'_i]$, where pwd_i is n-bit long and b_i is two-bit long. If $b_i = 01$, we call e'_i a *guess query*, which indicates that DEC should reveal the next bit in the password. (When $b_i = 10$, the adversary can use pwd_i to cause **B** to reveal whether or not the message piece is all zeros.)

(1) **(Integrity Check)** Check if $\beta'_i = G_k(\bar{i}\alpha'_i)$; if not, output FAIL;

(2) otherwise, perform the following action:

   (a) **(Password)** if $b_i = 01$, then let j be the number of guess queries that DEC has seen so far; if $j < n$ and the j^{th} bit of $F_k(\bar{0})$ is 0, output FAIL;

   (b) **(Answer to the Challenge)** if $b_i = 10$ and $pwd_i = F_k(\bar{0})$ and $\alpha'_i \oplus F_k(\overline{i+1}) = \bar{0}$, output FAIL;

   (c) otherwise $m'_i = \alpha'_i \oplus F_k(\overline{i+1})$.

Claim 4.1.1. *Session protocol* $\Pi = (ENC, DEC)$ *satisfies the Definition of Privacy #1.*

Proof. Suppose that an adversary $C = \{C_n\}$ breaks the privacy of Π under the Definition of Privacy #1. We will construct an adversary D which breaks the pseudo-randomness of the underlying function generator F. Let q_C be the probability defined in the Definition of Privacy #1, and assume that there exists a constant c such that for infinitely many n, $q_C(n) > \frac{1}{2} + \frac{1}{n^c}$; fix such an n. We will construct an adversary D_n that breaks the pseudo-randomness of F on key length n.

We construct D_n as follows, given a function $f : \{0,1\}^n \longrightarrow \{0,1\}^n$:

D_n chooses a random k' for G. D_n simulates both **A** and C_n:

When C_n asks for the encrypted piece of m_i, D_n computes $\alpha_i = m_i \oplus f(\overline{i+1})$ and gives the encrypted piece $e_i = [\alpha_i, \bar{0}, 00, G_{k'}(\bar{i}\alpha_i)]$ to C_n.

When C_n chooses two pieces m^0 and m^1, D_n randomly chooses one, remembers its choice ch and gives C_n the encrypted piece of m^{ch}.

When C_n sends a sequence of strings $e'_0 e'_1 \dots e'_i \dots$ to **B**, for each of the pieces D_n simulates DEC as follows (let $e'_i = [\alpha'_i, pwd_i, b_i, \beta'_i]$):

(1) If $\beta'_i \neq G_{k'}(\bar{i}\alpha'_i)$, D_n gives FAIL to C_n;

(2) otherwise, perform the following:

  (a) If $b_i = 01$, let j be the number of guess queries D_n has seen so far and it checks if $j < n$ and the j^{th} bit of $f(\bar{0})$ is 0; if so, D_n gives FAIL to C_n; otherwise it gives "NOT FAIL" to C_n.

  (b) If $b_i = 10$ and $pwd_i = f(\bar{0})$, D_n checks if $\alpha'_i \oplus f(\overline{i+1}) = \bar{0}$; if so, it gives FAIL to C_n.

  (c) In all other cases, D_n gives "NOT FAIL" to C_n.

Finally, when C_n outputs a guess at ch, D_n accepts if it is equal to ch, otherwise D_n rejects.

When f is a truly random function, if C_n correctly guesses the value of $f(\bar{0})$ before it sends a string to **B**, it can use $f(\bar{0})$ to learn which piece was encrypted and therefore its success at guessing ch is 1. Suppose that l is the number of message pieces C_n chooses in total, and the adversary has at most l chances to guess $f(\bar{0})$ by choosing a message to be $\bar{0}$ for **A**. Therefore the probability that the adversary guesses $f(\bar{0})$ correctly is $< l \cdot \frac{1}{2^n}$. In all other cases, the only information C_n sees about ch is $m^{ch} \oplus f(\overline{i+1})$ and no other encrypted piece uses $f(\overline{i+1})$. Hence the adversary learns no information about ch. Also note that no information about f is revealed in step (1), and so receiving a FAIL due to step (1) will not help C_n. Therefore the overall probability that $ch' = ch$ is $< \frac{1}{2} + l \cdot \frac{1}{2^n} \cdot \frac{1}{2}$.

When f is F_k for a random k, the probability that D_n accepts is $p(n) > \frac{1}{2} + \frac{1}{n^c}$; and when f is a truly random function, the probability is $< \frac{1}{2} + l \cdot \frac{1}{2^n} \cdot \frac{1}{2}$. Therefore the difference is at least $\frac{1}{n^c} - \frac{l}{2^{n+1}} > \frac{1}{n^{c'}}$, for some integer c'.

□

Claim 4.1.2. *Session protocol* $\Pi = (ENC, DEC)$ *satisfies the Definition of Integrity #3.*

Proof. Suppose that an adversary C breaks the integrity of Π under the Definition of Integrity #3. We will construct an adversary D which breaks the pseudo-randomness of the underlying function generator G. Let p_C be the probability defined in the Definition of Integrity #3, and assume that there exists a constant c such that for infinitely many n, $p_C(n) > \frac{1}{n^c}$; fix such an n.

Suppose we are given a function $g : \{0,1\}^{2n} \longrightarrow \{0,1\}^n$. D_n chooses a random key k for F. Then D_n simulates $\mathbf{A}$ and C_n as follows:

When C_n chooses m_i, D_n computes $\alpha_i = m_i \oplus F_k(\overline{i+1})$ gives the encrypted piece $e_i = [\alpha_i, \bar{0}, 00, g(\bar{i}\alpha_i)]$ to C_n.

When C_n sends a sequence of strings $e'_0 e'_1 \dots e'_i \dots$ to $\mathbf{B}$, for each of the pieces D_n simulates DEC as follows (let $e'_i = [\alpha'_i, pwd_i, b_i, \beta'_i]$):

(1) If $\beta'_i \neq g(\bar{i}\alpha'_i)$, D_n gives FAIL to C_n;

(2) otherwise, perform the following:

  (a) If $b_i = 01$, let j be the number of guess queries D_n has seen so far and it checks if $j < n$ and the j^{th} bit of $F_k(\bar{0})$ is 0; if so, D_n gives FAIL to C_n.

  (b) If $b_i = 10$ and $pwd_i = F_k(\bar{0})$, D_n checks if $\alpha'_i \oplus F_k(\overline{i+1}) = \bar{0}$; if so, it gives FAIL to C_n.

  (c) In all other cases, D_n computes $m'_i = \alpha'_i \oplus F_k(i+1)$ and gives "NOT FAIL" to C_n.

Let l be the number of message pieces C_n chooses for $\mathbf{A}$ and l' be the number of (supposedly) encrypted pieces C_n sends to $\mathbf{B}$. D_n will accept g if there is an i, $0 \leq i \leq l'$, such that one of the following statment is true: 1)$i > l$; 2) $i \leq l$, $\beta'_i = g(\bar{i}\alpha'_i)$, $m_i \neq m'_i$ and $m'_i \neq$ FAIL.

Since D_n runs a perfect simulation of the security experiment for C_n, a pseudo-randomly generated g will be accepted with probability $> \frac{1}{n^c}$.

Since step (2) does not reveal any information about g, any responses produced in step (2) will not help the adversary. Therefore if g is randomly generated, it will only be accepted if C_n guesses the value of g on an input it did not query. Since C_n has l' chances to guess the value of g, its probability of success is $< l' \cdot \frac{1}{2^n}$.

□

Claim 4.1.3. *Session protocol $\Pi = (ENC, DEC)$ does not satisfy the Definition of Privacy #2, i.e. there exists an adversary that breaks the privacy of Π under the Definition of Privacy #2.*

Proof. We will show how to construct such an adversary D. Specifically, we will construct D_n for any n.

D_n chooses $r = n+1$; that is, it will try to guess the $(n+1)$-th message piece.

D_n chooses $m_0, m_1, \dots, m_{n-1}$ arbitrarily and sees their encrypted pieces of the form $[\alpha_i, \bar{0}, \bar{0}, \beta_i]$.

Then D_n chooses $\bar{0}$ and $\bar{1}$ and sees the encrypted piece of one of them $e_r = [\alpha_r, \bar{0}, \bar{0}, \beta_r]$.

Then D_n sends $e'_0, e'_1, \dots, e'_{n-1}$ where $e'_j = [\alpha_j, \bar{0}, \bar{1}, \beta_j]$ for $j = 0, 1, \dots, n-1$, and learns each bit of $F_k(\bar{0})$ from outputs of DEC.

Then D_n sends $[\alpha_i, F_k(\bar{0}), \bar{0}, \beta_i]$ to $\mathbf{B}$; if $\mathbf{B}$ output FAIL, D outputs $ch' = 0$; otherwise it outputs 1.

Since D_n always sends a correct message-signature pair to DEC, DEC will never output FAIL due to invalid β's. Therefore **probability**$(ch' = ch) = 1$ and D_n breaks the privacy of Π.

□

We simply combine all three claims to conclude the proof.

□

Theorem 4.2. *There exists a session protocol that satisfies the Definition of Privacy #2 and the Definition of Integrity #3, but does not satisfy the Definition of Privacy #3.*

Proof. The difference between Privacy #2 and #3 is that the adversary in Privacy #3 is allowed to choose more message pieces for **A** after it sends (supposedly) encrypted pieces to **B**. Therefore we need to construct a session protocol that can be broken if we let *ENC* reveal some secret information. In order to force the adversary to access *ENC* at least one more time after seeing some responses from **B**, we make a particular message piece into a *special message piece*. Similarly to the session protocol in Theorem 4.1, the adversary first send some (supposedly) encrypted pieces and learn the *special message piece* from the responses of **B** one bit at a time. Then the adversary chooses the *special message piece* for **A**, and *ENC* reveals a password in the encrypted piece of the *special message piece*. Finally, the adversary inserts the password in an encrypted piece and learn some information about the message piece that is encrypted, which helps with guessing whether $\bar{0}$ or $\bar{1}$ was encrypted as the challenge encrypted piece.

Consider the following session protocol $\Pi = (ENC, DEC)$:

Let F, G be pseudo-random function generators, where $F_k : \{0,1\}^n \longrightarrow \{0,1\}^n$, and $G_{k'} : \{0,1\}^{2n} \longrightarrow \{0,1\}^n$ for any integer k, k'. Let kk' be the session key, where k and k' are the n-bit keys for F and G, respectively.

For $z(n)$-bit message pieces $m_0, m_1, \ldots$, *ENC* encrypts each m_i as follows:

(1) **(Password)** If $m_i = F_k(\bar{0})$, $e_i = [\alpha_i, F_k(\bar{1}, 00, G_{k'}(\bar{i}\alpha_i)]$, where $\alpha_i = m_i \oplus F_k(\overline{i+2}))$;

(2) otherwise, $e_i = [\alpha_i, \bar{0}, 00, G_{k'}(\bar{i}\alpha_i)]$, where $\alpha_i = m_i \oplus F_k(\overline{i+2})$ and $\beta_i = G_{k'}(\bar{i}\alpha_i)$.

For a sequence of strings of length $3n+1$ $e'_0, e'_1, \ldots$, *DEC* decrypts each e'_i as follows:

Let $e'_i = [\alpha'_i, pwd_i, b_i, \beta'_i]$, where pwd_i is n-bit long and b_i is two-bit long. If $b_i = 01$, we call e'_i a *guess query*, which indicates that *DEC* should reveal the next bit in the password. (When $b_i = 10$, the adversary can use pwd_i to cause **B** to reveal whether or not the message piece is all zeros.)

(1) **(Integrity Check)** If $\beta'_i \neq G_{k'}(\bar{i}\alpha'_i)$, output FAIL;

(2) otherwise, perform the following:

   (a) **(Special Message Piece)** if $b_i = 01$, then let j be the number of guess queries that *DEC* has seen so far; if $j < n$ and the j^{th} bit of $F_k(\bar{0})$ is 0, output FAIL;

   (b) **(Answer to the Challenge)** if $b_i = 10$ and $pwd_i = F_k(\bar{1})$ and $\alpha'_i \oplus F_k(\overline{i+2}) = \bar{0}$, output FAIL;

   (c) otherwise $m'_i = \alpha'_i \oplus F_k(\overline{i+2})$.

Claim 4.2.1. *Session protocol* $\Pi = (ENC, DEC)$ *satisfies the Definition of Privacy #2.*

Proof. Suppose that an adversary $C = \{C_n\}$ breaks the privacy of Π under the Definition of Privacy #2. We will construct an adversary D which breaks the pseudo-randomness of the underlying function generator F. Let q_C be the probability defined in the Definition of Privacy #2, and assume that there exists a constant c such that for infinitely many n, $q_C(n) > \frac{1}{2} + \frac{1}{n^c}$; fix such an n. We will construct an adversary D_n that breaks the pseudo-randomness of F on key length n.

We construct D_n as follows, given a function $f : \{0,1\}^n \longrightarrow \{0,1\}^n$:

D_n chooses a random k' for G. When C_n asks for the encrypted piece of m_i, D_n simulates *ENC* as follows:

(1) D_n checks if $m_i = f(\bar{0})$; if so, it gives $[\alpha_i, f(\bar{1}), 00, G_{k'}(\bar{i}\alpha_i)]$ to C_n, where $\alpha_i = m_i \oplus f(\overline{i+2})$;

(2) otherwise it gives $[\alpha_i, \bar{0}, 00, \beta_i]$ to C_n, where $\alpha_i = m_i \oplus f(\overline{i+2})$ and $\beta_i = G_{k'}(\bar{i}\alpha_i)$.

When C_n chooses two pieces, D_n randomly chooses one, remembers its choice ch and gives C_n its encrypted piece (by simulating ENC in the same way as in the previous step).

When C_n sends a string e_i' to **B**, D_n simulates DEC as follows: (Let $e_i' = [\alpha_i', pwd_i, b_i, \beta_i']$.)

(1) If $\beta_i' \neq G_k(\bar{i}\alpha_i')$, D_n gives FAIL to D_n;

(2) otherwise, perform the following:

 (a) if $b_i = 01$, then let j be the number of guess queries that DEC has seen so far; if the j^{th} bit of $f(\bar{0})$ is 0, D_n gives FAIL to C_n;

 (b) if $b_i = 10$ and $pwd_i = f(\bar{1})$ and $\alpha_i' \oplus f(\overline{i+2}) = \bar{0}$, D_n gives FAIL to C_n;

 (c) in all other cases, D_n gives "NOT FAIL" to C_n.

Finally, when C_n outputs a guess at ch, D_n accepts if it is equal to ch, otherwise D_n rejects.

When f is a truly random function, if the adversary correctly guesses the value of $f(\bar{0})$ (before it is allowed to send stuff to B) or $f(\bar{1})$, its success at guessing ch is 1. This happens with probability $< 2l \cdot \frac{1}{2^n}$ where l is the number of message pieces and (supposedly) encrypted pieces C_n chooses for both **A** and **B** in total and is a polynomial of n. (the exact probability is lower than this because there are fewer chances of guessing the value of $f(\bar{0})$.) In all other cases, the only information the adversary sees about ch is $m^{ch} \oplus f(\overline{i+2})$ and no other encrypted piece uses $f(\overline{i+2})$. Hence the adversary learns no information about ch. Also note that no information about f is revealed in step (1), and so receiving a FAIL due to step (1) will not help C_n. Therefore the overall probability that $ch' = ch$ is $< \frac{1}{2} + \frac{2l}{2^n} \cdot \frac{1}{2}$.

When f is F_k for a random k, the probability that D_n accepts is $p(n) > \frac{1}{2} + \frac{1}{n^c}$; and when f is a truly random function, the probability is $< \frac{1}{2} + \frac{2l}{2^n} \cdot \frac{1}{2}$. Therefore the difference is at least $\frac{1}{n^c} - \frac{1}{2^n} - \frac{2l}{2^{n+1}} > \frac{1}{n^{c'}}$, for some integer c'. □

Claim 4.2.2. *Session protocol $\Pi = (ENC, DEC)$ satisfies the Definition of Integrity #3.*

Proof. Suppose that an adversary $C = \{C_n\}$ breaks the integrity of Π under the Definition of Integrity #3. We will construct an adversary D break the pseudo-randomness of the underlying function generator G. Let p_C be the probability defined in the Definition of Integrity #3, and assume that there exists a constant c such that for infinitely many n, $p_C(n) > \frac{1}{n^c}$; fix such an n.

Suppose we are given a function $g : \{0,1\}^{2n} \longrightarrow \{0,1\}^n$. D_n chooses key k for F. Then D_n simulates **A** and C_n as follows:

When C_n asks for the encrypted piece of m_i, D_n simulates ENC as follows:

(1) D_n checks if $m_i = F_k(\bar{0})$; if so, it gives $[\alpha_i, F_k(\bar{1}), 00, g(\bar{i}\alpha_i)]$ to C_n, where $\alpha_i = m_i \oplus F_k(\overline{i+2})$;

(2) otherwise it gives $[\alpha_i, \bar{0}, 00, \beta_i]$ to C_n, where $\alpha_i = m_i \oplus F_k(\overline{i+2})$ and $\beta_i = G_{k'}(\bar{i}\alpha_i)$.

When C_n sends a string e_i' to **B**, D_n simulates DEC as follows: (Let $e_i' = [\alpha_i', pwd_i, b_i, \beta_i']$.)

(1) If $\beta_i' \neq g(\bar{i}\alpha_i')$, D_n gives FAIL to C_n;

(2) otherwise, perform the following:

   (a) if $b_i = 01$, then let j be the number of guess queries that DEC has seen so far; if $j < n$ and the j^{th} bit of $F_k(\bar{0})$ is 0, D_n gives FAIL to C_n;

   (b) if $b_i = 10$ and $pwd_i = F_k(\bar{1})$ and $\alpha'_i \oplus F_k(\overline{i+2}) = \bar{0}$, D_n gives FAIL to C_n;

   (c) in all other cases, D_n computes $m'_i = \alpha'_i \oplus F_k(\overline{i+2})$ and gives "NOT FAIL" to C_n.

At any point during the experiment, let l be the number of message pieces C_n has chosen for **A** and l' be the number of (supposedly) encrypted pieces C_n has sent to **B**. D_n will accept g if there is an i, $0 \leq i \leq l'$, such that one of the following statment is true: 1)$i > l$; 2) $i \leq l$, $\beta'_i = g(\bar{i}\alpha'_i)$, $m_i \neq m'_i$ and $m'_i \neq$ FAIL.

Since D_n runs a perfect simulation of the security experiment for C_n, a pseudo-randomly generated g will be accepted with probability $> \frac{1}{n^c}$.

Since step (2) does not reveal any information about g, any responses produced in step (2) will not help the adversary. Therefore if g is randomly generated, it will only be accepted if C_n guesses the value of g on an input it did not query. Therefore its probability of success is $< l \cdot \frac{1}{2^n}$.

□

Claim 4.2.3. *Session protocol* $\Pi = (ENC, DEC)$ *does not satisfy the Definition of Privacy #3, i.e. there exists an adversary that breaks the privacy of* Π *under the Definition of Privacy #3.*

Proof. We will show how to construct such an adversary D. Specifically, we will construct D_n for any n.

D_n chooses $r = n + 1$; that is, it will try to guess the $(n+1)$-th message piece.

D_n chooses $m_0, m_1, \ldots, m_{n-1}$ arbitrarily and sees their encrypted pieces of the form $[\alpha_i, \bar{0}, 00, \beta_i]$.

Then D_n chooses $\bar{0}$ and $\bar{1}$ and sees the encrypted piece of one of them $e_r = [\alpha_r, \bar{0}, 00, \beta_r]$.

Then D_n sends $e'_0, e'_1, \ldots, e'_{n-1}$ where $e'_j = [\alpha'_j, \bar{0}, 01, \beta_j]$ for $j = 0, 1, \ldots, n-1$, and learns each bit of $F_k(\bar{0})$ from outputs of DEC.

Then D_n sends $F_k(\bar{0})$ to ENC and learns $F_k(\bar{1})$;

Then D_n sends $[\alpha_i, F_k(\bar{1}), 10, \beta_i]$ to **B**; if **B** outputs FAIL, D outputs $ch' = 0$; otherwise it outputs 1.

Since D_n always sends a correct message-signature pair to DEC, DEC will never output FAIL due to invalid β's. Therefore **probability**$(ch' = ch) = 1$ and D_n breaks the privacy of Π.

□

We simply combine both claims to conclude the proof of the theorem.

□

5 More Robust Notions of Session Protocol

There are various ways in which an error may occur during a session, and it would be ideal to reduce the loss of message pieces from those errors as much as possible. In a session protocol that is secure under our definition of security, if a (supposedly) encrypted piece is altered during transmission, which causes **B** to output a FAIL symbol, the subsequent (supposedly) encrypted pieces are still decrypted normally. Therefore our definition of session protocol has some resilience against this type of error. However, if two (supposedly) encrypted pieces swap positions when they arrive at **B**, **B** would output FAIL on both encrypted pieces. Similarly if a (supposedly) encrypted piece is dropped or a new encrypted piece is inserted, the subsequent encrypted pieces would arrive out of their original positions, which would cause **B** to output FAIL on all of them. Nonetheless, there are alternative definitions of session protocol that offer the ability to recover from those types of error.

We can grant session protocols the ability to recover from out-of-order pieces by asking the decrypting algorithm to output the indices of the decrypted pieces. Upon successful decryption of a (supposedly) encrypted piece, DEC outputs a pair (i, m) where i is the index of the decrypted piece. In the case of an unsuccessful decryption (i.e. DEC rejects), DEC outputs a special symbol FAIL as in our previous definition. The adversary is considered to have broken the integrity if an encrypted piece is decrypted into something different from the message piece with the same index in the input stream of message pieces. Under this definition of session protocols, if a message is out-of-order, duplicated or missing, the indices of the output would reflect the errors immediately. This definition also results in different definitions of integrity and privacy. In particular, the adversary now should be able to see the index of a message piece as a part of the response from **B**.

These definitions of session protocol provide protection for message pieces that arrive out of order, but it is still impossible to recover encrypted pieces on which **B** outputs FAIL symbols. Another change that we can make is to allow **B** to send responses to **A** so that **A** is able to learn whether message pieces have been delivered successfully. Again, the definition of session security would need to be modified appropriately because the adversary may be able to gain information from this additional phase of interaction between **A** and **B**. However, the information **B** sends to **A** is also controlled by the adversary. As a result, the information exchanged in this phase is also subject to the adversary to the same degree as in our previous definition. It would be interesting to propose a formal definition of session protocol in this setting, but we will omit it from this paper.

References

[1] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation. In *In Proceedings of 38th Annual Symposium on Foundations of Computer Sci ence (FOCS 97*, 1997.

[2] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology EUROCRYPT 2001*, volume 2045, pages 453–474, 2001.

[3] Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In *Advances in Cryptology EUROCRYPT 2002*, volume 2332, pages 337–351, 2002.

[4] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[5] Jonathan Katz and Moti Yung. Complete characterization of security notions for probabilistic private-key encryption. In *Journal of Cryptology*, pages 245–254. ACM Press, 2000.

[6] Michael Luby. *Pseudorandomness and Cryptographic Applications*. Princeton University Press, Princeton, NJ, USA, 1994.

[7] Chanathip Namprempre. Secure channels based on authenticated encryption schemes: A simple characterization. In *Advances in Cryptology ASIACRYPT 2002*, volume 2501, pages 111–118, 2002.

[8] Charles Rackoff. Lecture notes for introduction to cryptography. `http://www.cs.toronto.edu/~rackoff/2426f08/notes4.pdf`, 2008.