The Best of Both Worlds
Authenticated Encryption

Foundations of Cryptography
Computer Science Department
Wellesley College

Fall 2016

Table of contents

Secrecy and integrity

- We began our studies with techniques for obtaining *secrecy* and moved on to ensuring message integrity.
- Why not build systems that ensure both at all times?
- Well there is an old saying ...

Our goal

- We seek an "ideally secure" communication channel that provides both secrecy and integrity.
- Not so easy. Instead, we provide a simpler set of definitions that treat secrecy and integrity separately, which suffices to understand the key issues.
- We begin with a CCA-secure private-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Since Π does not satisfy the syntax of a message authentication code, we must introduce a definition specific to the case.

Unforgeable encryption schemes

The unforgeable encryption experiment Enc-forge$_{\mathcal{A},\Pi}(n)$:

1. A random key k is generated by running Gen(1^n).
2. The adversary $\mathcal{A}$ is given input 1^n and oracle access to Enc$_k(\cdot)$. The adversary eventually outputs a ciphertext c.
3. Let $m := \text{Dec}_k(c)$ and let $\mathcal{Q}$ denote the set of all queries that $\mathcal{A}$ asked its encryption oracle. The output of the experiment is defined to be 1 if and only if (1) $m \neq \perp$; and (2) $m \notin \mathcal{Q}$.

Definition 4.16. A private-key encryption scheme Π is *unforgeable* if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr[\text{Enc-forge}_{\mathcal{A},\Pi}(n) = 1] \leq \text{negl}(n).$$

Definition 4.17. A private-key encryption scheme is an *authenticated encryption scheme* if it is CCA-secure and unforgeable.

It may be tempting to think ...

- Any reasonable combination of a secure encryption scheme and a secure message authentication code should result in an authenticated encryption scheme.
- Not so fast, these things must be done delicately or you ruin the spell.
- Let's start with a CPA-secure encryption scheme $\Pi_E = (\text{Gen, End, Dec})$ and a message authentication code $\Pi_M = (\text{Mac, Vrfy})$.

*Encrypt-and-authenticate**

Encrypt-and-authenticate: Encryption and authentication are computed independently in parallel. That is given m, the sender transmits the ciphertext $\langle c, t \rangle$ where:

$$c \leftarrow \mathrm{Enc}_{k_E}(m) \text{ and } t \leftarrow \mathrm{Mac}_{k_M}(m)$$

The receiver decrypts c to recover m; assuming no error occurred, it then verifies the tag t. If $\mathrm{Vrfy}_{k_M}(m, t) = 1$ the receiver outputs m; otherwise it outputs an error.

*We analyze this and other schemes when they are instantiated with "generic" secure components, i.e., an arbitrary CPA-secure encryption and an arbitrary (strongly) secure MAC.

Analysis of encrypt-and-authenticate

- This approach may not achieve even the most basic level of secrecy, since a secure MAC does not guarantee *any* secrecy and so it it possible for the tag $\mathrm{Mac}_{k_M}(m)$ to leak information about m to an eavesdropper.
- In fact it is likely to be insecure against chosen-plaintext attacks even when instantiated with standard components.
- In particular, if a *deterministic* MAC like CBC-MAC is used, then the tag computed on a message is the same every time. This allows an eavesdropper to identify when the same message is sent twice.

Authenticate-then-encrypt

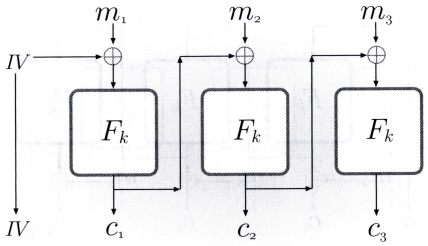
Authenticate-then-encrypt: Here a MAC tage t is first computed, and then the message and tag are encrypted together. That is, give a message m

$$t \leftarrow \mathrm{Mac}_{k_M}(m) \text{ and } c \leftarrow \mathrm{Enc}_{k_E}(m \parallel t).$$

The receiver decrypts c to recover $m \parallel t$; assuming no error occurred, it then verifies the tag t. As before, if $\mathrm{Vrfy}_{k_M}(m, t) = 1$ the receiver outputs m; otherwise it outputs an error.

Sad, but true

This one fails too, even for some of our old CPA-secure favorites such as CBC-mode-with-padding.

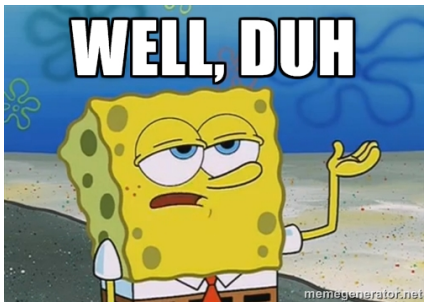


Recall this scheme works by first padding the plaintext (in our case $m \parallel t$) so the result is a multiple of the block length, and then encrypting the result using CBC mode.
There are two sources of potential decryption failure: A "bad-padding" error, the MAC tag does not verify.*

*Assuming attacker can distinguish between the two, she can now apply the same chosen-ciphertext attack described earlier.

Why not ensure there is only a single error message

- There may be legitimate reasons* to have multiple error messages.
- Forcing the error messages to be the same means that the combination is no longer truly generic.
- Most of all, it is extrordinarily hard to ensure that the different errors cannot be distinguished.**



*Usability, debugging, etc.

**Even a difference in the time to return each of these errors may be used to distinguish them.

Encrypt-then-authenticate

Encrypt-then-authenticate: In this case, the message m is first encrypted to obtain c and then a MAC tag t is computed over the result. The ciphertext is a pair $\langle c, t \rangle$. That is, given a message m

$$c \leftarrow \mathrm{Enc}_{k_E}(m) \text{ and } t \leftarrow \mathrm{Mac}_{k_M} c.$$

If $\mathrm{Vrfy}_{k_M}(c, t) = 1$ the receiver decrypts c and outputs the result; otherwise it outputs an error.

A generic construction of an authenticate encryption scheme

Construction 4.18. Let $\Pi_E = (\text{Gen}, \text{Enc}, \text{Dec})$ be a private-key encryption scheme and let $\Pi_M = (\text{Mac}, \text{Vrfy})$ be a message authentication code, where each key is a uniformly chosen n-bit value. Define a private-key encryption scheme (Gen', Enc', Dec') as follows:

- Gen': On input a key (k_E, k_M) and plaintext m, compute
- Enc': On input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^n$, compute $c \leftarrow \text{Enc}_{k_E}(m)$ and $t \leftarrow \text{Mac}_{k_M} c$. Output the ciphertext $\langle c, t \rangle$.
- Dec': On input a key (k_E, k_M) and ciphertext $\langle c, t \rangle$, first check whether $\text{Vrfy}_{k_M}(c, t) \stackrel{?}{=} 1$. If yes, then output $\text{Dec}_{k_E}(c)$; if no, then output $\perp$.

Party time: This approach is sound

Strong security ensures that the adversary will be unable to generate any valid ciphertext that it did not receive from its encryption oracle, so the scheme is unforgeable.

The MAC computed over the ciphertext has the effect of rendering the decryption oracle useless, since for every ciphertext $\langle c, t \rangle$ submitted to the decryption oracle, either

1. The adversary already knows the decryption because it received $\langle c, t \rangle$ from its encryption oracle; or
2. The adversary will almost surely get an error since the adversary cannot generate any new, valid ciphertexts.

This means CCA-security of the combined scheme reduces to the CPA-security of Π_E.

A real live authenticated encryption scheme

Theorem 4.19. Let $\Pi_E = (\text{Gen}, \text{Enc}, \text{Dec})$ be a CPA secure private-key encryption scheme and let $\Pi_M = (\text{Mac}, \text{Vrfy})$ be a a strongly secure message authentication code, then Construction 4.18 is an authenticated encryption scheme.
Proof. let Π' denote the scheme resulting from Construction 4.18. We need to show that Π' is unforgeable, and that it is CCA-secure.

Call a ciphertext $\langle c, t \rangle$ *valid* if $\text{Vrfy}_{k_M}(c, t) = 1$. We show that strong security of Π_M implies that (except with negligible probability) any "new"* ciphertexts the adversary submits to the decryption oracle will be invalid.

Let $\mathcal{A}$ be a PPT adversary attacking Construction 4.18 in a chosen-ciphertext attack and let *ValidQuery* be the event that $\mathcal{A}$ submits a new valid ciphertext to its decryption oracle.

*Here, *new* means that $\mathcal{A}$ did not receive $\langle c, t \rangle$ from its encryption oracle or as the challenge ciphertext.

Security against chosen-ciphertext attacks (CCA) revisited

The CCA indistinguishability experiment $\text{PrivK}^{\text{cca}}_{\mathcal{A},\Pi'}(n)$:

1. A key k is generated by running $\text{Gen}(1^n)$.
2. The adversary $\mathcal{A}$ is given 1^n and oracle access to $\text{Enc}_k(\cdot)$ and $\text{Dec}_k(\cdot)$. It outputs a pair of messages $m_0, m_1 \in \mathcal{M}$ of the same length.
3. A random bit $b \leftarrow \{0, 1\}$ is chosen. A *challenge ciphertext* $c \leftarrow \text{Enc}_k(m_b)$ is computed and given to $\mathcal{A}$.
4. The adversary $\mathcal{A}$ continues to have oracle access to $\text{Enc}_k(\cdot)$ and $\text{Dec}_k(\cdot)$, but is not allowed to query the latter on the challenge ciphertext. Eventually $\mathcal{A}$ outputs a bit b'.
5. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise. We write $\text{PrivK}^{\text{eav}}_{\mathcal{A},\Pi'}(n) = 1$ if the output is 1 and in this case we say that $\mathcal{A}$ *succeeded*.

Off we go then

Claim 4.20. Pr[ValidQuery] is negligible.
Proof. Define $\mathcal{A}_M$ attacking Π_M in experiment Mac-sforge$_{\mathcal{A}_M,\Pi_M}(n)$:
Adversary $\mathcal{A}_M$: $\mathcal{A}_M$ is given 1^n and access to oracle $\text{Mac}_{k_M}(\cdot)$.

1. Choose uniform $k_E \in \{0,1\}^n$ and $i \in \{1,...,q(n)\}$*.
2. Run $\mathcal{A}$ on input 1^n. When $\mathcal{A}$ makes an encryption-oracle query for m, answer as follows:
   2.1 Compute $c \leftarrow \text{Enc}_{k_E}(m)$.
   2.2 Query c to the MAC oracle and receive t. Return $\langle c, t\rangle$ to $\mathcal{A}$.

   The challenge ciphertext is done the same way with $b \in \{0,1\}$ chosen to select m_b.

   When $\mathcal{A}_M$ makes a decryption-oracle query for $\langle c, t\rangle$, answer as follows: If this is the ith decryption-oracle query, output $\langle c, t\rangle$. Otherwise
   2.1 If $\langle c, t\rangle$ was a response to a previous encryption-oracle query for a message m, return m.
   2.2 Otherwise, return $\perp$.

Strong MACs revisited

The message authentication experiment Mac-sforge$_{\mathcal{A}_M,\Pi_M}(n)$:

1. A random key k is generated by running Gen(1^n).
2. The adversary $\mathcal{A}_M$ is given input 1^n and oracle access to $\text{Mac}_k(\cdot)$. The adversary eventually outputs a pair (m, t). Let $\mathcal{Q}$ denote the set of all pairs,(m, t) that $\mathcal{A}_M$ queried $\text{Mac}_k(m)$ and received tag t in response.
3. The output of the experiment is defined to be 1 if and only if (1) $\text{Vrfy}(m, t) = 1$; and (2) $(m, t) \notin \mathcal{Q}$.

Definition 4.3. A message authentication code $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$ is *strongly secure* if for all probabilistic polynomial-time adversaries $\mathcal{A}_M$ there exists a negligible function negl such that

$$\Pr[\text{Mac-sforge}_{\mathcal{A},\Pi}(n) = 1] \leq \text{negl}(n).$$

Probability that $\mathcal{A}_M$ produces a good forgery

- The view of $\mathcal{A}$ when run as a subroutine by $\mathcal{A}_M$ is distributed identically to the view of $\mathcal{A}$ in experiment $\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n)$ until event ValidQuery occurs.
- Thus, the probability of event ValidQuery in experiment $\text{Mac-sforge}_{\mathcal{A}_M,\Pi_M}(n)$ is the same as the probability of that event in experiment $\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n)$.
- If $\mathcal{A}_M$ guesses the first index when ValidQuery occurs, then If $\mathcal{A}_M$ outputs $\langle c, t\rangle$ for which $\mathsf{Vrfy}_{k_M}(c,t) = 1$ and $\mathcal{A}_M$ succeeds in $\text{Mac-sforge}_{\mathcal{A}_M,\Pi_M}(n)$. The probability of guessing i correctly is $1/q(n)$, so

$$\Pr[\text{Mac-sforge}_{\mathcal{A}_M,\Pi_M}(n) = 1] \geq \Pr[\text{ValidQuery}] \cdot \frac{1}{q(n)}.$$

  Since Π_M is strongly secure and q is a polynomial, we conclude Pr[ValidQuery] is negligible. □

Π' is unforgeable

- The adversary $\mathcal{A}'$ in the unforgeable encryption experiment has access only to an encryption oracle and so is a restricted version of the adversary in the chosen-ciphertext experiment.
- The authors of our text claim that $\mathcal{A}'$ outputs a ciphertext $\langle c, t\rangle$, it "succeeds" only if $\langle c, t\rangle$ is valid and new, and that this is negligible by Claim 4.20. Frankly this makes no sense to me.
- Instead, we can use $\mathcal{A}'$ that attacks $\text{Enc-Forge}_{\mathcal{A}',\Pi'}(n)$ in place of $\mathcal{A}$ to construct an adversary $\mathcal{A}_M$ as before. This time no need for challenge ciphtertext and $\mathcal{A}'$ makes no decryption-oracle queries, but when it halts and outputs its pair $\langle c, t\rangle$, so does $\mathcal{A}_M$.

CCA-Security

We must show that Π' is CCA-secure. Let $\mathcal{A}$ be a PPT adversary attacking Construction 4.18 in a chosen-ciphertext attack. We have

$$\begin{aligned}&\Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1] \\ &\quad \leq \quad \Pr[\mathsf{ValidQuery}] + \Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1 \wedge \overline{\mathsf{ValidQuery}}]\end{aligned}$$

We have already shown the first term is negligible. Time for another claim:

Claim 4.21 There exists a function negl such that

$$\Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1 \wedge \overline{\mathsf{ValidQuery}}] \leq \frac{1}{2} + \mathsf{negl}(n).$$

To finish the proof ...

Proof Define $\mathcal{A}_E$ attacking Π_E in a chosen-plaintext attack:
Adversary $\mathcal{A}_M$: $\mathcal{A}_M$ is given 1^n and access to oracle $\mathsf{Enc}_{k_E}(\cdot)$.

1. Choose uniform $k_M \in \{0,1\}^n$.
2. Run $\mathcal{A}$ on input 1^n. When $\mathcal{A}$ makes an encryption-oracle query for m, answer as follows:
   - 2.1 Query m to $\mathsf{Enc}_{k_E}(\cdot)$ and receive c.
   - 2.2 Compute $t \leftarrow \mathsf{Mac}_{k_M}(c)$ and return $\langle c, t\rangle$ to $\mathcal{A}$.

   When $\mathcal{A}_M$ makes a decryption-oracle query for $\langle c, t\rangle$, answer as follows:
   - If $\langle c, t\rangle$ was a response to a previous encryption-oracle query for message m, return m. Otherwise return $\perp$.
3. When $\mathcal{A}$ outputs message m_0, m_1, output these same message and receive a challenge ciphertext c in response. Compute $t \leftarrow \mathsf{Mac}_{k_M}(c)$, and return $\langle c, t\rangle$ as the challenge ciphertext for $\mathcal{A}$.
4. Output the same bit b' that is output by $\mathcal{A}$.

In conclusion

The view of $\mathcal{A}$ when run as a subroutine by $\mathcal{A}_E$ is distributed identically to the view of $\mathcal{A}$ in experiment $\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n)$ as long as ValidQuery never occurs.
Thus, the probability that $\mathcal{A}_E$ succeeds when ValidQuery does not occur is the same as the probability that $\mathcal{A}$ succeeds when ValidQuery does not occur:

$$\Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A}_E,\Pi_E}(n) = 1 \wedge \overline{\mathsf{ValidQuery}}] = \Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1 \wedge \overline{\mathsf{ValidQuery}}]$$

implying that

$$\begin{aligned} \Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A}_E,\Pi_E}(n) = 1] &\geq \Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A}_E,\Pi_E}(n) = 1 \wedge \overline{\mathsf{ValidQuery}}] \\ &= \Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1 \wedge \overline{\mathsf{ValidQuery}}] \end{aligned}$$

□

Important safety tip

Basic Cryptographic Principle. Different instances of cryptographic primitives should always use independent keys.

Object Lesson. Suppose F (and therefore F^{-1}) is a strong pseudorandom permutation. Define $\mathsf{Enc}_{k_1}(m) = F_{k_1}(m \parallel r)$ for $m \in \{0,1\}^{n/2}$ and a uniform $r \in \{0,1\}^{n/2}$, and define $\mathsf{Mac}_{k_2}(c) = F^{-1}_{k_2}(c)$.

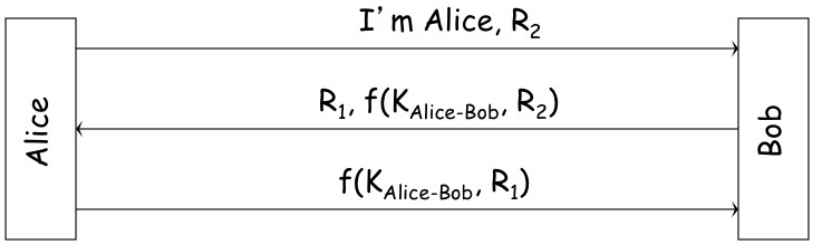
It can be shown that this encryption scheme is CPA-secure, and we know that given message code is a secure MAC. However, the encrypt-then-authenticate combination using the same key k applied to m yields:

$$\mathsf{Enc}_k(m), \mathsf{Mac}_k(\mathsf{Enc}_k(m)) = F_k(m \parallel r), F^{-1}_k(F_k(m \parallel r)) = F_k(m \parallel r), m \parallel r,$$

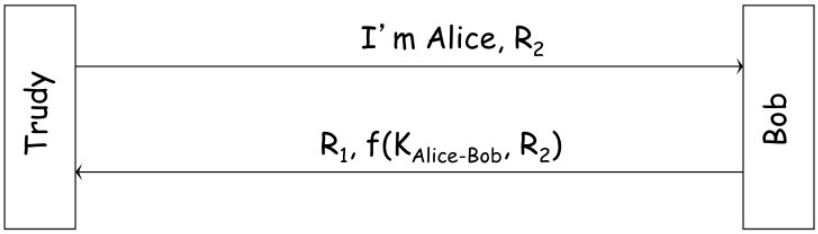
and the message m is revealed clear.

Another, slightly more convincing, example

A mutual authentication scheme based on a shared secret:



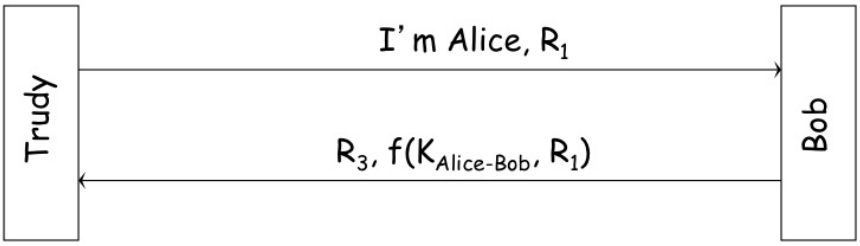
Reflection attack: Trudy wants to impersonate Alice to Bob



"I can't explain myself, I'm afraid sir," said Alice, "because, I'm not myself, you see."

Alice in Wonderland

Reflection attack: Trudy opens a second session Bob



*Which she still cannot complete. However, . . .