Symmetric-key cryptography VII

Message authentication codes (MACs)

Tanja Lange
(with lots of slides by Daniel J. Bernstein)

Eindhoven University of Technology

2MMC10 – Cryptology

Authenticated encryption

We have seen

- Alice and Bob use Diffie–Hellman to obtain a shared key.
- Bob is sure he is talking to Alice because he verified her signature.
- Alice can encrypt to Bob with symmetric crypto and vice versa.

Authenticated encryption

We have seen

- Alice and Bob use Diffie–Hellman to obtain a shared key.
- Bob is sure he is talking to Alice because he verified her signature.
- Alice can encrypt to Bob with symmetric crypto and vice versa.
- But Eve can mess with the ciphertext!
  With a stream cipher F, key k, and nonce n:
  $\text{Enc}(k, m, n) = F(k, n) \oplus m \quad \text{Dec}(k, c, n) = F(k, n) \oplus c$, thus

Authenticated encryption

We have seen

- Alice and Bob use Diffie–Hellman to obtain a shared key.
- Bob is sure he is talking to Alice because he verified her signature.
- Alice can encrypt to Bob with symmetric crypto and vice versa.
- But Eve can mess with the ciphertext!
  With a stream cipher F, key k, and nonce n:
  $\mathrm{Enc}(k, m, n) = F(k, n) \oplus m \quad \mathrm{Dec}(k, c, n) = F(k, n) \oplus c$, thus

  $$\mathrm{Dec}(k, \mathrm{Enc}(k, m, n), n) = F(k, n) \oplus F(k, n) \oplus m = m$$

Authenticated encryption

We have seen

- Alice and Bob use Diffie–Hellman to obtain a shared key.
- Bob is sure he is talking to Alice because he verified her signature.
- Alice can encrypt to Bob with symmetric crypto and vice versa.
- But Eve can mess with the ciphertext!
  With a stream cipher F, key k, and nonce n:
  $\mathrm{Enc}(k, m, n) = F(k, n) \oplus m \quad \mathrm{Dec}(k, c, n) = F(k, n) \oplus c$, thus
  $$\mathrm{Dec}(k, \mathrm{Enc}(k, m, n), n) = F(k, n) \oplus F(k, n) \oplus m = m$$
  $\mathrm{Dec}(k, \mathrm{Enc}(k, m, n) \oplus e, n) = F(k, n) \oplus F(k, n) \oplus m \oplus e = m \oplus e$

Authenticated encryption

We have seen

- Alice and Bob use Diffie–Hellman to obtain a shared key.
- Bob is sure he is talking to Alice because he verified her signature.
- Alice can encrypt to Bob with symmetric crypto and vice versa.
- But Eve can mess with the ciphertext!
  With a stream cipher F, key k, and nonce n:
  $\mathrm{Enc}(k, m, n) = F(k, n) \oplus m \quad \mathrm{Dec}(k, c, n) = F(k, n) \oplus c$, thus

  $$\mathrm{Dec}(k, \mathrm{Enc}(k, m, n), n) = F(k, n) \oplus F(k, n) \oplus m = m$$

  $$\mathrm{Dec}(k, \mathrm{Enc}(k, m, n) \oplus e, n) = F(k, n) \oplus F(k, n) \oplus m \oplus e = m \oplus e$$

  Need integrity protection.

Authenticated encryption

We have seen

- Alice and Bob use Diffie–Hellman to obtain a shared key.
- Bob is sure he is talking to Alice because he verified her signature.
- Alice can encrypt to Bob with symmetric crypto and vice versa.
- But Eve can mess with the ciphertext!
  With a stream cipher F, key k, and nonce n:
  $\mathrm{Enc}(k, m, n) = F(k, n) \oplus m \quad \mathrm{Dec}(k, c, n) = F(k, n) \oplus c$, thus
  $$\mathrm{Dec}(k, \mathrm{Enc}(k, m, n), n) = F(k, n) \oplus F(k, n) \oplus m = m$$
  $$\mathrm{Dec}(k, \mathrm{Enc}(k, m, n) \oplus e, n) = F(k, n) \oplus F(k, n) \oplus m \oplus e = m \oplus e$$
  Need integrity protection.
- Eve can also send ciphertexts, won't what they decrypt to.

Authenticated encryption

We have seen

- Alice and Bob use Diffie–Hellman to obtain a shared key.
- Bob is sure he is talking to Alice because he verified her signature.
- Alice can encrypt to Bob with symmetric crypto and vice versa.
- But Eve can mess with the ciphertext!
  With a stream cipher F, key k, and nonce n:
  $\mathrm{Enc}(k, m, n) = F(k, n) \oplus m \quad \mathrm{Dec}(k, c, n) = F(k, n) \oplus c$, thus
  $$\mathrm{Dec}(k, \mathrm{Enc}(k, m, n), n) = F(k, n) \oplus F(k, n) \oplus m = m$$
  $$\mathrm{Dec}(k, \mathrm{Enc}(k, m, n) \oplus e, n) = F(k, n) \oplus F(k, n) \oplus m \oplus e = m \oplus e$$
  Need integrity protection.
- Eve can also send ciphertexts, won't what they decrypt to.
  Need authenticator.

Authenticated encryption

We have seen

- Alice and Bob use Diffie–Hellman to obtain a shared key.
- Bob is sure he is talking to Alice because he verified her signature.
- Alice can encrypt to Bob with symmetric crypto and vice versa.
- But Eve can mess with the ciphertext!
  With a stream cipher F, key k, and nonce n:
  $\mathrm{Enc}(k, m, n) = F(k, n) \oplus m \quad \mathrm{Dec}(k, c, n) = F(k, n) \oplus c$, thus
  $$\mathrm{Dec}(k, \mathrm{Enc}(k, m, n), n) = F(k, n) \oplus F(k, n) \oplus m = m$$
  $$\mathrm{Dec}(k, \mathrm{Enc}(k, m, n) \oplus e, n) = F(k, n) \oplus F(k, n) \oplus m \oplus e = m \oplus e$$
  Need integrity protection.
- Eve can also send ciphertexts, won't what they decrypt to.
  Need authenticator.
- Message authentication codes achieve both.

Simple example of authentication code

Fix prime $p = 1000003$.

Assume sender knows independent uniform random secrets
$r_1, r_2, r_3, r_4, r_5 \in \{0, 1, \ldots, 999999\}$,
$s_1, s_2, \ldots, s_{100} \in \{0, 1, \ldots, 999999\}$,
Assume receiver knows the same secrets $r_1, r_2, \ldots, r_5, s_1, \ldots, s_{100}$.

Simple example of authentication code

Fix prime $p = 1000003$.

Assume sender knows independent uniform random secrets
$r_1, r_2, r_3, r_4, r_5 \in \{0, 1, \ldots, 999999\}$,
$s_1, s_2, \ldots, s_{100} \in \{0, 1, \ldots, 999999\}$,
Assume receiver knows the same secrets $r_1, r_2, \ldots, r_5, s_1, \ldots, s_{100}$.

Sender can now authenticate 100 ciphertexts $c_1, \ldots, c_{100}$,
each c_i having 5 components $c_{i,1}, c_{i,2}, c_{i,3}, c_{i,4}, c_{i,5}$
with $c_{i,j} \in \{0, 1, \ldots, 999999\}$.

Simple example of authentication code

Fix prime $p = 1000003$.

Assume sender knows independent uniform random secrets
$r_1, r_2, r_3, r_4, r_5 \in \{0, 1, \ldots, 999999\}$,
$s_1, s_2, \ldots, s_{100} \in \{0, 1, \ldots, 999999\}$,
Assume receiver knows the same secrets $r_1, r_2, \ldots, r_5, s_1, \ldots, s_{100}$.

Sender can now authenticate 100 ciphertexts $c_1, \ldots, c_{100}$,
each c_i having 5 components $c_{i,1}, c_{i,2}, c_{i,3}, c_{i,4}, c_{i,5}$
with $c_{i,j} \in \{0, 1, \ldots, 999999\}$.

Sender transmits $c_i = (c_{i,1}, c_{i,2}, c_{i,3}, c_{i,4}, c_{i,5})$ together with an authentication tag

$$t_i = (c_{i,1} r_1 + c_{i,2} r_2 + \cdots + c_{i,5} r_5 \bmod p) + s_i \bmod 1000000$$

and the message number i.

A MAC using fewer secrets

Instead of choosing independent
$r_1, r_2, \ldots, r_5, s_1, \ldots, s_{100}$,
choose $r, s_1, s_2, \ldots, s_{100}$.

A MAC using fewer secrets

Instead of choosing independent
$r_1, r_2, \ldots, r_5, s_1, \ldots, s_{100}$,
choose $r, s_1, s_2, \ldots, s_{100}$.

Sender transmits $c_i = (c_{i,1}, c_{i,2}, c_{i,3}, c_{i,4}, c_{i,5})$ together with an authentication tag

$$t_i = (c_{i,1} r^5 + c_{i,2} r^4 + \cdots + c_{i,5} r \bmod p) + s_i \bmod 1000000$$

and the message number i.

I.e.: take $r_i = r^{6-i}$ in previous

$$t_i = (c_{i,1} r_1 + c_{i,2} r_2 + \cdots + c_{i,5} r_5 \bmod p) + s_i \bmod 1000000$$

Compute via Horner's rule as

$$t_i = (((c_{i,1} r + c_{i,2}) r \cdots + c_{i,5}) r \bmod p) + s_i \bmod 1000000.$$

Security analysis

Attacker can observe several (c_i, t_i, i) tuples.
Attacker's goal: Find i', c', t' such that $c' \neq c_{i'}$ but
$t' = (c'(r) \bmod p) + s_{i'} \bmod 1000000$.
Here $c'(x) = \sum_i c'_j x^{6-j}$.

Security analysis

Attacker can observe several (c_i, t_i, i) tuples.
Attacker's goal: Find i', c', t' such that $c' \neq c_{i'}$ but
$t' = (c'(r) \bmod p) + s_{i'} \bmod 1000000$.
Here $c'(x) = \sum_i c'_j x^{6-j}$.

Obvious attack: Choose any $c' \neq c_1$. Choose uniform random t'.
Success chance $1/1000000$.

Security analysis

Attacker can observe several (c_i, t_i, i) tuples.
Attacker's goal: Find i', c', t' such that $c' \neq c_{i'}$ but
$t' = (c'(r) \bmod p) + s_{i'} \bmod 1000000$.
Here $c'(x) = \sum_i c'_j x^{6-j}$.

Obvious attack: Choose any $c' \neq c_1$. Choose uniform random t'.
Success chance $1/1000000$.

Each forgery attempt has chance $1/1000000$ of being accepted.

Security analysis

Attacker can observe several (c_i, t_i, i) tuples.
Attacker's goal: Find i', c', t' such that $c' \neq c_{i'}$ but
$t' = (c'(r) \bmod p) + s_{i'} \bmod 1000000$.
Here $c'(x) = \sum_i c'_j x^{6-j}$.

Obvious attack: Choose any $c' \neq c_1$. Choose uniform random t'.
Success chance $1/1000000$.

Each forgery attempt has chance $1/1000000$ of being accepted.

More subtle attack: Choose $c' \neq c_1$ so that the polynomial
$c'(x) - c_1(x)$ has 5 distinct roots $x \in \{0, 1, \ldots, 999999\}$
modulo p. Choose $t' = t_1$.

Security analysis

Attacker can observe several (c_i, t_i, i) tuples.
Attacker's goal: Find i', c', t' such that $c' \neq c_{i'}$ but
$t' = (c'(r) \bmod p) + s_{i'} \bmod 1000000$.
Here $c'(x) = \sum_i c'_j x^{6-j}$.

Obvious attack: Choose any $c' \neq c_1$. Choose uniform random t'.
Success chance $1/1000000$.

Each forgery attempt has chance $1/1000000$ of being accepted.

More subtle attack: Choose $c' \neq c_1$ so that the polynomial
$c'(x) - c_1(x)$ has 5 distinct roots $x \in \{0, 1, \ldots, 999999\}$
modulo p. Choose $t' = t_1$.

E.g. $c_1 = (0, 0, 0, 0, 100)$, $c' = (1, 0, 0, 1, 125)$:
$c'(x) - c_1(x) = x^5 + x^2 + 25x$ which has five roots mod p:
$0, 299012, 334447, 631403, 735144$.

Security analysis

Attacker can observe several (c_i, t_i, i) tuples.
Attacker's goal: Find i', c', t' such that $c' \neq c_{i'}$ but
$t' = (c'(r) \bmod p) + s_{i'} \bmod 1000000$.
Here $c'(x) = \sum_i c'_j x^{6-j}$.

Obvious attack: Choose any $c' \neq c_1$. Choose uniform random t'.
Success chance $1/1000000$.

Each forgery attempt has chance $1/1000000$ of being accepted.

More subtle attack: Choose $c' \neq c_1$ so that the polynomial
$c'(x) - c_1(x)$ has 5 distinct roots $x \in \{0, 1, \ldots, 999999\}$
modulo p. Choose $t' = t_1$.

E.g. $c_1 = (0, 0, 0, 0, 100)$, $c' = (1, 0, 0, 1, 125)$:
$c'(x) - c_1(x) = x^5 + x^2 + 25x$ which has five roots mod p:
$0, 299012, 334447, 631403, 735144$.

Success chance $5/1000000$.

Small nitpick

Actually, success chance can be above $5/1000000$.

Remember: $t_i = (c_{i,1}r^5 + \cdots + c_{i,5}r \bmod p) + s_i \bmod 1000000$.

Small nitpick

Actually, success chance can be above $5/1000000$.
Remember: $t_i = (c_{i,1}r^5 + \cdots + c_{i,5}r \bmod p) + s_i \bmod 1000000$.

If $c_1(334885) \bmod p \in \{1000000, 1000001, 1000002\}$
then a forgery $(c', t_1, 1)$ with $c'(x) = c_1(x) + x^5 + x^2 + 25x$ also succeeds for r;
success chance $6/1000000$.

Reason: 334885 is a root of $x^5 + x^2 + 25x + 1000000 = (x + 665118)(x^2 + 82920x + 84624)(x^2 + 251965x + 667039)$.

Small nitpick

Actually, success chance can be above $5/1000000$.
Remember: $t_i = (c_{i,1}r^5 + \cdots + c_{i,5}r \bmod p) + s_i \bmod 1000000$.

If $c_1(334885) \bmod p \in \{1000000, 1000001, 1000002\}$
then a forgery $(c', t_1, 1)$ with $c'(x) = c_1(x) + x^5 + x^2 + 25x$ also succeeds for r;
success chance $6/1000000$.

Reason: 334885 is a root of $x^5 + x^2 + 25x + 1000000 =$
$(x + 665118)(x^2 + 82920x + 84624)(x^2 + 251965x + 667039)$.

Can have as many as 15 roots of
$(c'(x) - c_1(x)) \cdot (c'(x) - c_1(x) + 1000000) \cdot (c'(x) - c_1(x) - 1000000)$.

Small nitpick Bigger nitpick?

Actually, success chance can be above $5/1000000$.
Remember: $t_i = (c_{i,1}r^5 + \cdots + c_{i,5}r \bmod p) + s_i \bmod 1000000$.

If $c_1(334885) \bmod p \in \{1000000, 1000001, 1000002\}$
then a forgery $(c', t_1, 1)$ with $c'(x) = c_1(x) + x^5 + x^2 + 25x$ also succeeds for r;
success chance $6/1000000$.

Reason: 334885 is a root of $x^5 + x^2 + 25x + 1000000 =$
$(x + 665118)(x^2 + 82920x + 84624)(x^2 + 251965x + 667039)$.

Can have as many as 15 roots of
$(c'(x)-c_1(x))\cdot(c'(x)-c_1(x)+1000000)\cdot(c'(x)-c_1(x)-1000000)$.

Can attacker improve chance by picking different values for t'?

Small nitpick Bigger nitpick?

Actually, success chance can be above $5/1000000$.
Remember: $t_i = (c_{i,1}r^5 + \cdots + c_{i,5}r \bmod p) + s_i \bmod 1000000$.

If $c_1(334885) \bmod p \in \{1000000, 1000001, 1000002\}$
then a forgery $(c', t_1, 1)$ with $c'(x) = c_1(x) + x^5 + x^2 + 25x$ also succeeds for r;
success chance $6/1000000$.

Reason: 334885 is a root of $x^5 + x^2 + 25x + 1000000 =$
$(x + 665118)(x^2 + 82920x + 84624)(x^2 + 251965x + 667039)$.

Can have as many as 15 roots of
$(c'(x) - c_1(x)) \cdot (c'(x) - c_1(x) + 1000000) \cdot (c'(x) - c_1(x) - 1000000)$.

Can attacker improve chance by picking different values for t'?

No. Every choice of (c', t', i') with $c' \neq c_{i'}$ has chance $\leq 15/1000000$ of being accepted by receiver because $(c'(x) - c_1(x) - t' + t_1) \cdot (c'(x) - c_1(x) - t' + t_1 + 10^6) \cdot (c'(x) - c_1(x) - t' + t_1 - 10^6)$ has ≤ 15 roots modulo p.

Poly1305

Poly1305 uses 128-bit r's, with 22 bits cleared for speed.

Computes modulo $p = 2^{130} - 5$. Adds s_i mod 2^{128}.

Generate r and s_i from master key and i. Never reuse for same i.

Poly1305

Poly1305 uses 128-bit r's, with 22 bits cleared for speed.
Computes modulo $p = 2^{130} - 5$. Adds $s_i \bmod 2^{128}$.
Generate r and s_i from master key and i. Never reuse for same i.

For $\leq L$-byte messages, each forgery succeeds for $\leq 8\lceil L/16 \rceil$ choices of r. Probability $\leq 8\lceil L/16 \rceil / 2^{106}$.

Poly1305

Poly1305 uses 128-bit r's, with 22 bits cleared for speed.
Computes modulo $p = 2^{130} - 5$. Adds $s_i \bmod 2^{128}$.
Generate r and s_i from master key and i. Never reuse for same i.

For $\leq L$-byte messages, each forgery succeeds for $\leq 8 \lceil L/16 \rceil$ choices of r. Probability $\leq 8 \lceil L/16 \rceil / 2^{106}$.

D forgeries are all rejected with probability $\geq 1 - 8D \lceil L/16 \rceil / 2^{106}$,
e.g. 2^{64} forgeries, $L = 1536$: $\Pr[\text{all rejected}] \geq 0.9999999998$.

Poly1305

Poly1305 uses 128-bit r's, with 22 bits cleared for speed.
Computes modulo $p = 2^{130} - 5$. Adds s_i mod 2^{128}.
Generate r and s_i from master key and i. Never reuse for same i.

For $\leq L$-byte messages, each forgery succeeds for $\leq 8 \lceil L/16 \rceil$ choices of r. Probability $\leq 8 \lceil L/16 \rceil / 2^{106}$.

D forgeries are all rejected with probability $\geq 1 - 8D \lceil L/16 \rceil / 2^{106}$, e.g. 2^{64} forgeries, $L = 1536$: $\Pr[\text{all rejected}] \geq 0.9999999998$.

Handle variable-length c, ensure that different $c(x)$ are different polynomials mod p.

Poly1305

Poly1305 uses 128-bit r's, with 22 bits cleared for speed.
Computes modulo $p = 2^{130} - 5$. Adds $s_i \bmod 2^{128}$.
Generate r and s_i from master key and i. Never reuse for same i.

For $\leq L$-byte messages, each forgery succeeds for $\leq 8\lceil L/16 \rceil$ choices of r. Probability $\leq 8\lceil L/16 \rceil / 2^{106}$.

D forgeries are all rejected with probability $\geq 1 - 8D\lceil L/16 \rceil / 2^{106}$, e.g. 2^{64} forgeries, $L = 1536$: Pr[all rejected] ≥ 0.9999999998.

Handle variable-length c, ensure that different $c(x)$ are different polynomials mod p.

Split c_i into 16-byte blocks, maybe with smaller final block; append 1 to each block; view as little-endian integers $\bar{c}_{i,j}$.
For all but last block $\bar{c}_{i,j} \in \{2^{128}, 2^{128} + 1, \ldots, 2^{129} - 1\}$.

Poly1305

Poly1305 uses 128-bit r's, with 22 bits cleared for speed.
Computes modulo $p = 2^{130} - 5$. Adds $s_i \bmod 2^{128}$.
Generate r and s_i from master key and i. Never reuse for same i.

For $\leq L$-byte messages, each forgery succeeds for $\leq 8\lceil L/16\rceil$ choices of r. Probability $\leq 8\lceil L/16\rceil/2^{106}$.

D forgeries are all rejected with probability $\geq 1 - 8D\lceil L/16\rceil/2^{106}$,
e.g. 2^{64} forgeries, $L = 1536$: Pr[all rejected] ≥ 0.9999999998.

Handle variable-length c, ensure that different $c(x)$ are different polynomials mod p.

Split c_i into 16-byte blocks, maybe with smaller final block;
append 1 to each block; view as little-endian integers $\bar{c}_{i,j}$.
For all but last block $\bar{c}_{i,j} \in \{2^{128}, 2^{128}+1, \ldots, 2^{129}-1\}$.

$$t_i = (\bar{c}_{i,1}r^k + \bar{c}_{i,2}r^{k-1} + \cdots + \bar{c}_{i,k}r \bmod 2^{130}-5) + s_i \bmod 2^{128},$$

where $k = \lceil L_i/16\rceil$ for c_i of L_i bytes.