CS4501 Cryptographic Protocols
Lecture 21: Digital Signatures, Dolev-Strong Broadcast

https://jackdoerner.net/teaching/#2026/Spring/CS4501

Roadmap for the Last Few Lectures

Lecture 19, Theorem 2: In the setting where $t < n/2$, there exists a t-secure BA protocol if an only if there exists a t-secure broadcast protocol.

Lecture 20, Theorem 1: When $t \geq n/3$, there does not exist a t-secure BA protocol in the plain model, even assuming authenticated channels.

Lecture 20, Theorem 2: When $t < n/3$, there exists a perfectly t-secure BA protocol in the plain model, assuming authenticated channels.

Lecture 21, Theorem 2: Assuming the existence of one-way functions and a *public-key infrastructure*, there exists a (computationally) t-secure broadcast protocol in the setting where $t < n$.

Lecture 22, Theorem 2: Any t-secure *deterministic* broadcast protocol requires at least $t + 1$ rounds.

The PKI Model

- In this class, we will use two slightly different versions of *the PKI model.*
- Informally, we will assume that every party begins the experiment knowing the public keys of all other parties, and that all keys have been sampled appropriately.
- When we need to formalize this, we will use a simple *PKI functionality* $\mathcal{F}_{\text{PKI}}$ which accepts a key from one party and transmits it reliably to all of the others.
- This is exactly the same as the broadcast functionality, except for its name. In the literature, there are other kinds of PKI functionalities, but we won't worry about them in class.
- Thus, when we construct broadcast in the PKI model, we're effectively doing *broadcast extension.*
- OT extension turns κ OT instances into polynomially many OT instances. Broadcast extension turns n broadcasts (one for each party, at the beginning of the experiment) into polynomially many broadcasts.

Digital Signature Schemes

What do we want from *digital signatures*? Let's develop some intuition first:

- **Authentication:** Bob knows that Alice generated a message, even if the message was delivered by an *un*authenticated channel or an untrusted third party.
- **Integrity:** Bob knows that nobody has tampered with the message in transit.
- **Transferability:** Without involving Alice, Bob can forward the signed message to Carol, who will *also* be convinced that Alice generated the message.
- **Non-Repudiation:** Alice cannot claim that she didn't sign the message.

Sometimes we might want additional properties, or we might not want all of these, but these seem to match our intuition about what physical signatures are supposed to guarantee. Next, we need to formalize them in the form of a game.

Defining Signatures

Definition 1 (Signature Syntax). A signature scheme Σ for a message space $\mathcal{M}$ is a trio of PPT algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ such that:

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\kappa)$ samples a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(m)$ samples a signature σ given a message $m \in \mathcal{M}$ and the secret key sk.
- $\mathsf{Verify}_{\mathsf{pk}}(m, \sigma)$ outputs 1 if and only if σ is a valid signature on m under pk.

Definition 2 (Signature Correctness). A signature scheme Σ for $\mathcal{M}$ is *correct* if there is a negligible function ε such that for every $m \in \mathcal{M}$,

$$\Pr\left[\begin{array}{l}\mathsf{Verify}_{\mathsf{pk}}(m, \sigma) = 0 : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa), \\ \qquad\qquad\qquad\qquad \sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(m)\end{array}\right] < \varepsilon(\kappa)$$

Defining Security for Signatures

Definition 1 (Signature Syntax). A signature scheme Σ for a message space $\mathcal{M}$ is a trio of PPT algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ such that:

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\kappa)$ samples a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(m)$ samples a signature σ given a message $m \in \mathcal{M}$ and the secret key sk.
- $\mathsf{Verify}_{\mathsf{pk}}(m, \sigma)$ outputs 1 if and only if σ is a valid signature on m under pk.

Definition 3 (Unforgeability). A signature scheme Σ for $\mathcal{M}$ is *Existentially Unforgeable under Chosen Message Attacks* (EUF-CMA Secure) if for every PPT $\mathcal{A}$ there is a negligible ε such that

$$\Pr\left[\mathsf{EUFCMAGame}_{\Sigma,\mathcal{A}}(\kappa) = 1\right] < \varepsilon(\kappa)$$

$\mathsf{EUFCMAGame}_{\Sigma,\mathcal{A}}(\kappa)$

$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\kappa)$

$Q := \emptyset$

$1^\kappa, \mathsf{pk}$ →

← $m \in \mathcal{M}$

$Q := Q \cup \{m\}$

$\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(m)$

σ →

← m^*, σ^*

repeat until satisfied

outputs 1 if

$\mathsf{Verify}_{\mathsf{pk}}(m^*, \sigma^*) = 1$

$\wedge\ m^* \notin Q$

outputs 0 otherwise

Question: *what happens if the challenger doesn't insist* $m^* \notin Q$*?*

Defining Security for Signatures

Definition 3 (Unforgeability). A signature scheme Σ for $\mathcal{M}$ is *Existentially Unforgeable under Chosen Message Attacks* (EUF-CMA Secure) if for every PPT $\mathcal{A}$ there is a negligible ε such that

$$\Pr\left[\mathsf{EUFCMAGame}_{\Sigma,\mathcal{A}}(\kappa) = 1\right] < \varepsilon(\kappa)$$

Definition 4 (ℓ-time Unforgeability). A signature scheme Σ for $\mathcal{M}$ is *ℓ-Time* EUF-CMA Secure if for every PPT $\mathcal{A}$ there is a negligible ε such that

$$\Pr\left[\mathsf{EUFCMAGame}_{\Sigma,\mathcal{A}}(\kappa) = 1 \,\middle|\, |Q| \leq \ell(\kappa)\right] < \varepsilon(\kappa)$$

Question: *these definitions says it's hard for $\mathcal{A}$ to forge a signature on some m^* that was* never *signed under* pk. *Suppose $\mathcal{A}$ already saw a valid σ^* on m^*. Is it hard to find a* second *valid signature $\sigma^\dagger \neq \sigma^*$?*

$\mathsf{EUFCMAGame}_{\Sigma,\mathcal{A}}(\kappa)$

$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\kappa)$

$Q := \emptyset$

$1^\kappa, \mathsf{pk}$ →

← $m \in \mathcal{M}$

$Q := Q \cup \{m\}$

$\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(m)$

σ →

← m^*, σ^*

repeat until satisfied

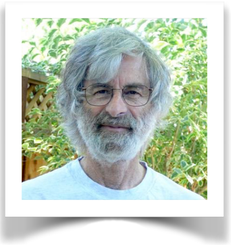
outputs 1 if

$\mathsf{Verify}_{\mathsf{pk}}(m^*, \sigma^*) = 1$

$\wedge\, m^* \notin Q$

outputs 0 otherwise

Lamport's One-Time Signatures



Definition 5 (One-Time Signatures for $\mathcal{M} = \{0,1\}^\ell$, for some constant ℓ):
Let f be a one-way function. The scheme comprises the following algorithms:

- $\mathsf{Gen}(1^\kappa, \ell)$ outputs $(\mathsf{sk}, \mathsf{pk})$ where $\forall i \in [\ell], \forall b \in \{0,1\}, x_i^b \leftarrow \{0,1\}^\kappa$ and $y_i^b := f(x_i^b)$, and where $\mathsf{sk} := \{(x_1^0, x_1^1), \ldots, (x_\ell^0, x_\ell^1)\}$ and $\mathsf{pk} := \{(y_1^0, y_1^1), \ldots, (y_\ell^0, y_\ell^1)\}$.
- $\mathsf{Sign}_{\mathsf{sk}}(m)$ outputs $\sigma_1 \| \ldots \| \sigma_\ell$ where $\sigma_i = x_i^{m_i}\ \forall i \in [\ell]$ and $m_1 \| \ldots \| m_\ell = m$.
- $\mathsf{Verify}_{\mathsf{pk}}(m_1 \| \ldots \| m_\ell, \sigma_1 \| \ldots \| \sigma_\ell)$ outputs 1 if and only if $\bigwedge_{i \in [\ell]} f(\sigma_i) = y_i^{m_i}$.

Theorem 1: If f is a OWF and ℓ is polynomially-bounded with respect to κ, then Lamport's scheme is correct and one-time EUF-CMA secure.

Proof: Perfect correctness follows from the fact that f is deterministic. We will show that any adversary who forges a signature (after seeing one example) can be used to invert f.

Lamport's One-Time Signatures

Proof: Perfect correctness follows from the fact that f is deterministic. We will show that any adversary who forges a signature (after seeing one example) can be used to invert f.

In some instance of the one-time EUF-CMA game, let $\{m'\} = Q$ and $m'_1 \| \ldots \| m'_\ell = m'$.

Any $\mathcal{A}$ that wins the signature game must do so by outputting σ_i^* such that $f(\sigma_i^*) = y_i^{1-m'_i}$ for some $i \in [\ell]$. We will use $\mathcal{A}$ to construct a second adversary $\mathcal{A}'$ for the OWF game.

In the OWF game, $\mathcal{A}'$ receives some $\hat{y} \in \{f(x) : x \in \{0,1\}^\kappa\}$ as input. $\mathcal{A}'$ guesses $j \leftarrow [\ell]$ and $\mu \leftarrow \{0,1\}$ and then emulates an instance of the EUF-CMA game towards $\mathcal{A}$ in which $y_j^\mu := \hat{y}$, but otherwise behaves exactly as the challenger usually should.

If $\mathcal{A}'$ guesses correctly (i.e. $j = i \wedge \mu = 1 - m'_i$) and $\mathcal{A}$ succeeds in forging (which $\mathcal{A}'$ can verify), then $\mathcal{A}'$ outputs σ_j in the OWF game.

If succeeds $\mathcal{A}$ in forging with probability $\delta(\kappa)$, then $\mathcal{A}'$ succeeds in inverting the OWF with probability $\delta(\kappa)/(2\ell)$, which is negligible if is $\delta(\kappa)$ is negligible and ℓ is bounded by a polynomial in κ. ∎

Extending Lamport Signatures

Lamport Signatures have at least two apparent disadvantages:

- Each key can only be used with one message.
- For messages of length ℓ, both the key and the signature have length $\Omega(\ell \cdot \kappa)$.

It turns out that we can improve both of these with one-way functions alone.

Suppose we had a function family $\{h_k : \{0,1\}^* \to \{0,1\}^{|k|}\}_{k \in \{0,1\}^*}$ with the property that for any two-part PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible ε such that

$$\Pr\left[\begin{array}{r} x' \neq x \wedge h_k(x') = h_k(x) : (x, s) \leftarrow \mathcal{A}_1(1^\kappa), \\ k \leftarrow \{0,1\}^\kappa \\ x' \leftarrow \mathcal{A}_2(s, k) \end{array}\right] < \varepsilon(\kappa)$$

Extending Lamport Signatures

Suppose we had a function family $\{h_k : \{0,1\}^* \to \{0,1\}^{|k|}\}_{k \in \{0,1\}^*}$ with the property that for any two-part PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible ε such that

$$\Pr\left[\begin{array}{r} x' \neq x \wedge h_k(x') = h_k(x) : (x, s) \leftarrow \mathcal{A}_1(1^\kappa), \\ k \leftarrow \{0,1\}^\kappa \\ x' \leftarrow \mathcal{A}_2(s, k) \end{array}\right] < \varepsilon(\kappa)$$

This is a *Universal One-Way Hash Function*. We can prove that it exists iff OWFs do.

There are other kinds of *Hash Function* that are *not* known to be implied by OWFs. In CS6222, we will see that some are even separated from OWFs.

Finding a *second* preimage (that is, a second input that compresses to the same output as a first input) under a UOWHF is hard.

Extending Lamport Signatures

Suppose we enhance Lamport's signature scheme by sampling $k \leftarrow \{0,1\}^\kappa$ and adding k to the public key. Then we can use h_k to compress a message m of any polynomial length down to a κ-bit value m'. Then we sign m' instead of m.

Intuitively, this gives us $O(\kappa^2)$-length signatures for any polynomially-bounded ℓ.

How can we sign multiple messages?

Suppose we want to sign *two* messages, m_1 and m_2, using one key-pair $(\mathsf{sk}, \mathsf{pk})$.

Up front, we sample two auxiliary key-pairs $(\mathsf{sk}_1, \mathsf{pk}_1)$ and $(\mathsf{sk}_2, \mathsf{pk}_2)$, and compute $\hat{\sigma} \leftarrow \mathsf{Sign}_{\mathsf{sk}}(\mathsf{pk}_1 \| \mathsf{pk}_2)$. $\hat{\sigma}$ convinces anyone that sees it that pk_1 and pk_2 are *authorized.*

To sign m_i for $i \in [2]$, compute $\hat{\sigma}_i \leftarrow \mathsf{Sign}_{\mathsf{sk}_i}(m_i)$ and output $\sigma_i := (\mathsf{pk}_1, \mathsf{pk}_2, \hat{\sigma}, \hat{\sigma}_i)$.

Extending Lamport Signatures

To sign *many* messages, we can apply this idea recursively.

For *every possible* compressed message $m' \in \{0,1\}^\kappa$, sample a key pair $(\mathsf{sk}_{m'}, \mathsf{pk}_{m'})$. Arrange these public keys into pairs, and sign each pair of public keys $\mathsf{pk}_{m'_1} \| \mathsf{pk}_{m'_2}$ under *another* key-pair $(\mathsf{sk}_{m'_1 \| m'_2}, \mathsf{pk}_{m'_1 \| m'_2})$. Do this recursively, forming a tree of depth κ, until a single root public key is reached.

Now to sign a message of any polynomial length, our signature is of size $O(\kappa^3)$.

But we will have a problem: the overall secret key is exponentially large! It includes sub-keys for every possible $m' \in \{0,1\}^\kappa$. Fortunately, we can fix this using a PRF.

Every time we would sample a random value in the above scheme, we instead evaluate a PRF $F_{k'}$ at some deterministic index. Now our only secret is the PRF key $k' \leftarrow \{0,1\}^\kappa$. We can construct the relevant branch of the tree lazily, when we need it.

Beyond Lamport Signatures.

Proving that these extensions to Lamport signatures are actually secure is out of scope for this class. In any case, the final signature size of $\Theta(\kappa^3)$ is too big to use in practice.In practice, we usually use *short* signatures of length $O(\kappa)$ that are constructed using *structured assumptions* such as CDH, or the famous RSA assumption.

Digital signatures were introduced but *not* constructed in the Diffie-Hellman paper.

Frustratingly, constructing short signatures from the original Diffie-Hellman family of assumptions has long required either very special *pairing* groups in which CDH is thought to be hard but DDH is known to be easy, or else a special functionality called a *Random Oracle* which was proved by Goldwasser and Kalai to be unrealizable.

This was the best we could do until a short signature scheme from DDH *alone* was discovered by Catalano, Frasca, and Giunta … in March of 2026!

There are still many important open problems. Sometimes it takes 50 years to solve one. Maybe you could do it next :)

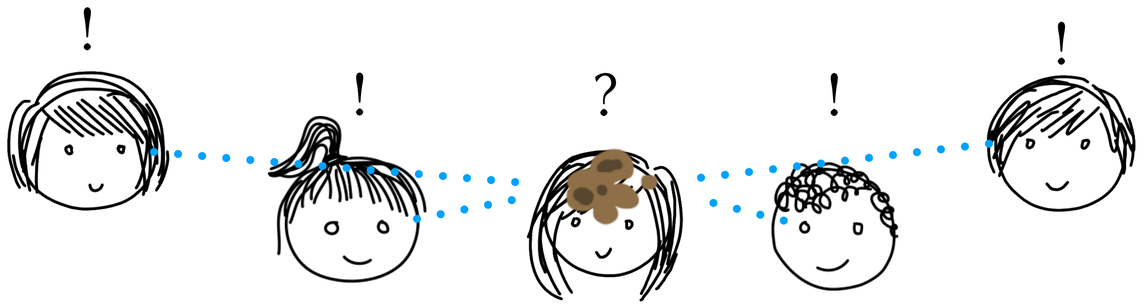
A Warmup: the Muddy Children Puzzle

Before we show how to construct Broadcast in the PKI model, let's develop some intuition about how a group of parties can start with distinct views and arrive at common knowledge by induction.

Consider a group of children that go outside to play in the rain. While playing, some of them get mud on their foreheads. Their mother is angry about the mess they have made, and calls them to come inside.

"If you know there is mud on your forehead at this moment, step forward!" she says.

A Warmup: the Muddy Children Puzzle



“If you know there is mud on your forehead at this moment, step forward!” she says.

Because the mother is very strict, the children cannot speak to one another. Each one can see whether the others have mud, but the children cannot see or feel the mud that might be on their own foreheads. *If only one child is muddy, how will that child know?*

A Warmup: the Muddy Children Puzzle

"If you know there is mud on your forehead at this moment, step forward!" she says.

Because the mother is very strict, the children cannot speak to one another. Each one can see whether the others have mud, but the children cannot see or feel the mud that might be on their own foreheads. *If only one child is muddy, how will that child know?*

If *two* children are muddy, then nobody steps forward initially. Their mother tries again. "Second call: everyone who knows they have mud on their forehead, step forward!" *How can both muddy children know who they are, now that she asked twice?*

Suppose three children are muddy. How many times does their mother need to ask?

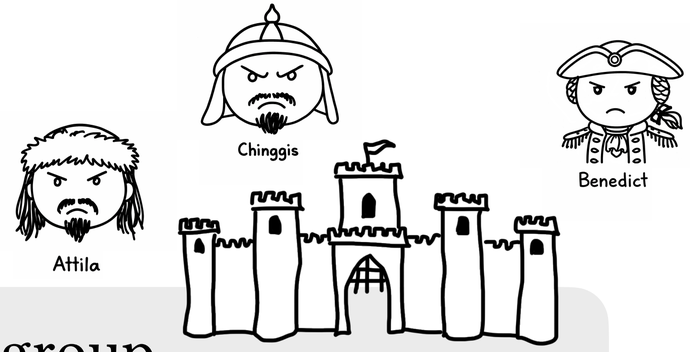
A Warmup: the Muddy Children Puzzle

The children in this puzzle arrive at common knowledge by reasoning counterfactually about how the other children would have behaved at a particular time, if they had made a particular observation about the world.

The number of rounds they need to narrow down the possibilities is determined by the number of children that are muddy.

Although the details are different, we will use a similar kind of reasoning to construct a broadcast protocol that works even when a majority of the children are dishonest.

Recall: BG/Broadcast



Definition 1. Let $n, t \in \mathbb{N}$ such that $t < n$, and let $P_1, \ldots, P_n$ be a group of parties that are connected by synchronous, authenticated channels. Let π be a protocol in which P_1 has an input $x \in \{0,1\}$ and every P_i produces an output $y_i \in \{0,1\}$. We say that π is a t-secure *Broadcast* protocol if the following conditions hold with all-but-negligible probability when $\mathcal{A}$ maliciously corrupts up to t parties:

1. **Consistency:** All honest parties output the same bit y.

2. **Validity:** If P_1 is honest, then $y = x$.

A broadcast *functionality* simply takes an input from one party and outputs it to all others. In other words, a simulation-secure broadcast protocol is one that securely computes $f(m, \lambda, \ldots, \lambda) = (m, m, \ldots, m)$.

Dolev-Strong: Simplifying Signatures

Before describing our broadcast protocol, we'll make an important simplification:

We will assume for this proof that signatures are *ideal*. Specifically, we will assume that $\Pr\left[\mathsf{EUFCMAGame}_{\Sigma,\mathscr{A}}(\kappa) = 1\right] = 0$.

Note that signatures with this property are *impossible* to construct: $\mathscr{A}$ can always guess a forgery, and have a nonzero chance of guessing successfully. Effectively, we are imagining an adversary that is *incapable* of manipulating signature strings at all.

This style of idealization is called the *Dolev-Yao model*. It was used by Dolev and Strong in their original proof, and it remains common in the distributed computing literature. It tends to yield very clean, simple proofs.

Bridging the gap from this model to one in which proper EUF-CMA signatures are used is tricky! We'll talk about it after the proof is finished.

Dolev-Strong: PKI Setup Phase

We will look at a couple of "warm up" protocols on our way to the real Dolev-Strong protocol. In order to avoid cluttering our slides and our minds, we'll describe up front how the public-key infrastructure is initialized at the start of the protocol experiment.

Remember: these instructions happen at the beginning of the next three protocols!

Let $P_1, \ldots, P_n$ be a set of PPT parties that are connected by authenticated channels, let κ be a security parameter, and let $\mathcal{F}_{\mathsf{PKI}}(i)$ be a PKI functionality that enables P_i specifically to broadcast a public key to the others.

1. Every P_i for $i \in [n]$ samples $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$ and sends pk_i to $\mathcal{F}_{\mathsf{PKI}}(i)$.

2. Every P_i for $i \in [n]$ receives pk_j from $\mathcal{F}_{\mathsf{PKI}}(j)$ for $j \in [n] \backslash \{i\}$.

First Warm Up: $n = 3,\ t = 2$

Suppose that P_1 wishes to send $x \in \{0,1\}$ to P_2 and P_3. The protocol is:

1. P_1 signs $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}_1}(x)$ and sends (x, σ) to P_2 and P_3.
2. P_2 and P_3 receive the tuple (x, σ) from P_1, and forward it to one another.
3. If P_i for $i \in [3]$ observes exactly one message (x, σ) such that $\mathsf{Verify}_{\mathsf{pk}_1}(x, \sigma) = 1$, then it outputs x. Otherwise it outputs 0.

Validity: If P_1 is honest, it signed exactly one value of x. $\mathscr{A}$ cannot forge any other signature under pk_1 in our ideal signatures model.

Consistency: If P_1 is *honest*, it follows from validity. If *only* P_1 is corrupted, the other two output the same value because they exchange their messages. If P_1 and (WLOG) P_2 are corrupted, then consistency is trivial.

Second Warm Up: $n = 4,\ t = 2$

Suppose that P_1 wishes to send $x \in \{0,1\}$ to P_2, P_3, and P_4. The protocol is:

1. P_1 signs $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}_1}(x)$ and sends (x, σ) to P_2, P_3, and P_4.

2. P_2, P_3, and P_4 receive the tuple (x, σ) from P_1, and forward it to one another.

3. If P_i for $i \in [4]$ observes exactly one message (x, σ) such that $\mathsf{Verify}_{\mathsf{pk}_1}(x, \sigma) = 1$, then it outputs x. Otherwise it outputs 0.

Validity: As before.

Consistency? Suppose that P_1 and P_2 are corrupted.

In round 1, P_1 sets $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}_1}(1)$ and sends $(1,\sigma)$ to P_3, and P_4.

In round 2, P_2 sets $\sigma' \leftarrow \mathsf{Sign}_{\mathsf{sk}_1}(0)$ and sends $(0,\sigma')$ to P_3, but *not* P_4.

P_3 outputs 0 because it saw two different signed messages, but P_4 outputs 1!

What went wrong, and how do we fix it?

The adversary used the public key of the corrupted P_1 to inject an error in the *last round*, and there wasn't time for the honest parties to alert one another.

How to fix this? An intuition:

- If an honest party sees (x, σ) it will forward it to all other honest parties.
- Honest parties should only accept an output if they're sure at least one honest party saw it *before* the last round.
- **Q:** Nobody knows who is honest, so *how can we be sure at least one honest party saw a particular* (x, σ)*?* **A:** by insisting that at least $t + 1$ parties do.
- **Q:** *How can they all know how many others saw a particular* (x, σ)*?*
  A: If you see it, you sign it (and pass it along)!

The Dolev-Strong Broadcast Protocol

Theorem 2: Let $n, t \in \mathbb{N}$ such that $t < n$. Assuming ideal digital signatures and a public-key infrastructure, there exists a t-secure broadcast protocol in the presence of any PPT adversary.

Note: we don't need authenticated channels, because we can *build* authenticated channels from plain ones, using signatures and PKI. *Can you see how?*

Proof: Let π_{DS} be a protocol for parties $P_1, \ldots, P_n$, where the sender P_1 has input x:

1. Every P_i initializes an empty set $\mathcal{V}_i := \emptyset$.

2. Round 1: P_1 signs $\sigma_1 \leftarrow \mathsf{Sign}_{\mathsf{sk}_1}(x)$ and sends $(x, (1, \sigma_1))$ to everyone else.

The Dolev-Strong Broadcast Protocol

1. Every P_i initializes an empty set $\mathscr{V}_i := \emptyset$.

2. Round 1: P_1 signs $\sigma_1 \leftarrow \mathsf{Sign}_{\mathsf{sk}_1}(x)$ and sends $(x, (1,\sigma_1))$ to everyone else.

3. Round $r \in [2,t+1]$: For every $(\hat{x}, (j_1, \sigma_1), \ldots, (j_r, \sigma_r))$ that P_i received at the end of the last round, if $\hat{x} \notin \mathscr{V}_i$ and the following *well-formedness* conditions hold:

   - $j_1, \ldots, j_r$ are distinct values in $[n]$ such that $j_1 = 1$ and $i \notin \{j_1, \ldots, j_r\}$
   - $\mathsf{Verify}_{\mathsf{pk}_j}(x, \sigma) = 1$ for every $(j, \sigma) \in \{(j_1, \sigma_1), \ldots, (j_r, \sigma_r)\}$

   Then P_i signs $\sigma_{r+1} \leftarrow \mathsf{Sign}_{\mathsf{sk}_i}(\hat{x})$, sends $(\hat{x}, (j_1, \sigma_1), \ldots, (j_r, \sigma_r), (i, \sigma_{r+1}))$ to everyone, and adds $\hat{x}$ to $\mathscr{V}_i$. Note the outgoing message is *always* well-formed if P_i is honest.

4. After Round $t+1$: P_1 outputs its input x. If $\mathscr{V}_i = \{\hat{x}\}$ for some $\hat{x}$ (i.e. $|\mathscr{V}_i| = 1$), then P_i outputs $\hat{x}$. Otherwise, P_i outputs 0.

The Dolev-Strong Broadcast Protocol

Termination: π_{DS} always terminates in *exactly* $t + 1$ rounds.

Validity: If P_1 is honest, then it signs *exactly* one value x and every other honest party receives it. In our ideal signatures model, no forgeries are possible, which means that no well-formed $(\hat{x}, (j_1, \sigma_1), \ldots, (j_r, \sigma_r))$ such that $\hat{x} \neq x$ can be computed, so it must the the case that $\mathscr{V}_k = \{x\}$ for every $k \in [n]$.

Consistency:

- If an honest P_i received some *well-formed* $(\hat{x}, (j_1, \sigma_1), \ldots, (j_r, \sigma_r))$ such that $\hat{x} \notin \mathscr{V}_i$ in round $r < t$, then *every* party received $(\hat{x}, (j_1, \sigma_1), \ldots, (j_r, \sigma_r), (i, \sigma_{r+1}))$ in round $r + 1$, and we know that $\hat{x} \in \mathscr{V}_k$ for every $k \in [n]$ such that P_k is honest at the end.

The Dolev-Strong Broadcast Protocol

Consistency:

- If an honest P_i received some *well-formed* $(\hat{x}, (j_1, \sigma_1), \ldots, (j_r, \sigma_r))$ such that $\hat{x} \notin \mathscr{V}_i$ in round $r < t$, then *every* party received $(\hat{x}, (j_1, \sigma_1), \ldots, (j_r, \sigma_r), (i, \sigma_{r+1}))$ in round $r + 1$, and we know that $\hat{x} \in \mathscr{V}_k$ for every $k \in [n]$ such that P_k is honest at the end.

- If an honest P_i receives some well-formed $(\hat{x}, (j_1, \sigma_1), \ldots, (j_{t+1}, \sigma_{t+1}))$ at the end of round $t + 1$, then there *must* be some honest party P_h such that $h \in \{j_1, \ldots, j_{t+1}\}$.

  Because no forgeries are possible in our ideal signatures model, we know that P_h signed $\hat{x}$ and forwarded it (in a well-formed way) to all parties previously.

  Therefore it *must* be that $\hat{x} \in \mathscr{V}_k$ for every $k \in [n]$ such that P_k is honest at the end.

- Since honest parties always ignore an ill-formed $(\hat{x}, (j_1, \sigma_1), \ldots, (j_r, \sigma_r))$, it must be that any pair of honest parties P_i, P_h have $\mathscr{V}_i = \mathscr{V}_h$, and output the same thing. ∎

The Dolev-Strong Broadcast Protocol

Theorem 2: Let $n, t \in \mathbb{N}$ such that $t < n$. Assuming ideal digital signatures and a public-key infrastructure, there exists a t-secure broadcast protocol in the presence of any PPT adversary.

Notice that the proof did not require (nor did the theorem specify) static corruptions! This is the first protocol that we proved secure against *adaptive* corruptions.

Recall **Lemma 2** from Lecture 20: Let $t < n/2$. If there exists t-secure broadcast, then there exists t-secure BA

Corollary 1: Let $n, t \in \mathbb{N}$ such that $t < n/2$. Assuming ideal digital signatures and a public-key infrastructure, there exists a t-secure BA protocol in the presence of any PPT adversary.

Proof: by Theorem 2 and Lemma 2.

The Dolev-Strong Broadcast Protocol

What's next? We will answer three questions:

1. How can we replace the ideal signatures (which imply an unrealistic restriction on the adversary) with standard EUF-CMA signatures?

2. How can we realize the broadcast functionality (so that we can use our composition theorem to build other protocols on top of broadcast), instead of just achieving a property-based notion of broadcast?

3. Do we really need so many rounds?

1. Replacing Ideal Signatures

Dolev and Strong gave essentially the same proof that you just saw in 1983.

How can we replace ideal signatures with real ones? *Any ideas?*

One way might be to create some hybrid experiments with signature schemes that aren't quite real, and aren't quite ideal. One end of the chain of hybrids is the real world, and our existing proof applies to the opposite end. Each pair of experiments must be related to one another using a reduction. This strategy would work, but it would be a bit messy, and nobody ever wrote down the details.

Alternate idea: devise a proper ideal functionality $\mathcal{F}_{\text{sig}}$ for signatures. Recast Dolev-Strong in the $\mathcal{F}_{\text{sig}}$-hybrid model using the exact proof you already saw, prove that any EUF-CMA signature scheme realizes $\mathcal{F}_{\text{sig}}$, and then use the composition theorem!

This strategy was finally published in 2025…

There are still many important open problems. Sometimes it takes ~~50~~ 40 years to solve one. Maybe you could do it next :)

CS4501 Cryptographic Protocols
Lecture 21: Digital Signatures, Dolev-Strong Broadcast

https://jackdoerner.net/teaching/#2026/Spring/CS4501