Cryptography

Lecture 1

Cryptography

Lecture 1

Our first encounter with secrecy:
Secret-Sharing

Secrecy

Secrecy

- Cryptography is all about "controlling access to information"
  - Access to learning and/or influencing information

Secrecy

- Cryptography is all about "controlling access to information"
  - Access to learning and/or influencing information
- One of the aspects of access control is secrecy

A Game

A Game

- A "dealer" and two "players" Alice and Bob

A Game

- A "dealer" and two "players" Alice and Bob
- Dealer has a message, say two bits $m_1 m_2$

A Game

- A "dealer" and two "players" Alice and Bob
- Dealer has a message, say two bits $m_1 m_2$
- She wants to "share" it among the two players so that neither player by herself/himself learns anything about the message, but together they can find it

A Game

- A "dealer" and two "players" Alice and Bob
- Dealer has a message, say two bits m_1m_2
- She wants to "share" it among the two players so that neither player by herself/himself learns anything about the message, but together they can find it
- Bad idea: Give m_1 to Alice and m_2 to Bob

A Game

- A "dealer" and two "players" Alice and Bob
- Dealer has a message, say two bits m_1m_2
- She wants to "share" it among the two players so that neither player by herself/himself learns anything about the message, but together they can find it
- Bad idea: Give m_1 to Alice and m_2 to Bob
- Other ideas?

Sharing a bit

Sharing a bit

- To share a bit m, Dealer picks a uniformly random bit b and gives $a := m \oplus b$ to Alice and b to Bob

Sharing a bit

- To share a bit m, Dealer picks a uniformly random bit b and gives $a := m \oplus b$ to Alice and b to Bob
  - Bob learns nothing (b is a random bit)

Sharing a bit

- To share a bit m, Dealer picks a uniformly random bit b and gives a := m⊕b to Alice and b to Bob
  - Bob learns nothing (b is a random bit)
  - Alice learns nothing either: for each possible value of m (0 or 1), a is a random bit (0 w.p. ½, 1 w.p. ½)

Sharing a bit

- To share a bit m, Dealer picks a uniformly random bit b and gives a := m⊕b to Alice and b to Bob
  - Bob learns nothing (b is a random bit)
  - Alice learns nothing either: for each possible value of m (0 or 1), a is a random bit (0 w.p. ½, 1 w.p. ½)

m = 0 → (a,b) = (0,0) or (1,1)

m = 1 → (a,b) = (1,0) or (0,1)

Sharing a bit

- To share a bit m, Dealer picks a uniformly random bit b and gives a := m⊕b to Alice and b to Bob
  - Bob learns nothing (b is a random bit)
  - Alice learns nothing either: for each possible value of m (0 or 1), a is a random bit (0 w.p. ½, 1 w.p. ½)

    m = 0 → (a,b) = (0,0) or (1,1)

    m = 1 → (a,b) = (1,0) or (0,1)
    - Her view is independent of the message

Sharing a bit

- To share a bit m, Dealer picks a uniformly random bit b and gives a := m⊕b to Alice and b to Bob
  - Bob learns nothing (b is a random bit)
  - Alice learns nothing either: for each possible value of m (0 or 1), a is a random bit (0 w.p. ½, 1 w.p. ½)
    - Her view is independent of the message
  - Together they can recover m as a⊕b

m = 0 → (a,b) = (0,0) or (1,1)

m = 1 → (a,b) = (1,0) or (0,1)

Sharing a bit

- To share a bit m, Dealer picks a uniformly random bit b and gives a := m⊕b to Alice and b to Bob
  - Bob learns nothing (b is a random bit)
  - Alice learns nothing either: for each possible value of m (0 or 1), a is a random bit (0 w.p. ½, 1 w.p. ½)
    - Her view is independent of the message
  - Together they can recover m as a⊕b
- Multiple bits can be shared independently: as, $m_1m_2 = a_1a_2 \oplus b_1b_2$

m = 0 → (a,b) = (0,0) or (1,1)

m = 1 → (a,b) = (1,0) or (0,1)

Sharing a bit

- To share a bit m, Dealer picks a uniformly random bit b and gives $a := m \oplus b$ to Alice and b to Bob
  - Bob learns nothing (b is a random bit)
  - Alice learns nothing either: for each possible value of m (0 or 1), a is a random bit (0 w.p. ½, 1 w.p. ½)
    - Her view is independent of the message
  - Together they can recover m as $a \oplus b$

$m = 0 \rightarrow (a,b) = (0,0)$ or $(1,1)$

$m = 1 \rightarrow (a,b) = (1,0)$ or $(0,1)$

- Multiple bits can be shared independently: as, $m_1m_2 = a_1a_2 \oplus b_1b_2$
- Note: any one share can be chosen before knowing the message [why?]

Secrecy

Secrecy

- Is the message m really secret?

Secrecy

- Is the message m really secret?
- Alice or Bob can correctly find the bit m with probability ½, by randomly guessing

Secrecy

- Is the message m really secret?
- Alice or Bob can correctly find the bit m with probability ½, by randomly guessing
  - Worse, if they already know something about m, they can do better (Note: we didn't say m is uniformly random!)

Secrecy

- Is the message m really secret?
- Alice or Bob can correctly find the bit m with probability ½, by randomly guessing
  - Worse, if they already know something about m, they can do better (Note: we didn't say m is uniformly random!)
- But they could have done this without obtaining the shares

Secrecy

- Is the message m really secret?
- Alice or Bob can correctly find the bit m with probability ½, by randomly guessing
  - Worse, if they already know something about m, they can do better (Note: we didn't say m is uniformly random!)
- But they could have done this without obtaining the shares
  - The shares didn't leak any additional information to either party

Secrecy

- Is the message m really secret?
- Alice or Bob can correctly find the bit m with probability ½, by randomly guessing
  - Worse, if they already know something about m, they can do better (Note: we didn't say m is uniformly random!)
- But they could have done this without obtaining the shares
  - The shares didn't leak any additional information to either party
- Typical crypto goal: **preserving** secrecy

Secrecy

- Is the message m really secret?
- Alice or Bob can correctly find the bit m with probability ½, by randomly guessing
  - Worse, if they already know something about m, they can do better (Note: we didn't say m is uniformly random!)
- But they could have done this without obtaining the shares
  - The shares didn't leak any additional information to either party
- Typical crypto goal: **preserving** secrecy
  - View is independent of the message

Secrecy

- Is the message m really secret?
- Alice or Bob can correctly find the bit m with probability ½, by randomly guessing
  - Worse, if they already know something about m, they can do better (Note: we didn't say m is uniformly random!)
- But they could have done this without obtaining the shares
  - The shares didn't leak any additional information to either party
- Typical crypto goal: **preserving** secrecy
  - View is independent of the message
    - i.e., for all possible values of the message, the view is distributed the same way

Secrecy

- Is the message m really secret?
- Alice or Bob can correctly find the bit m with probability ½, by randomly guessing
  - Worse, if they already know something about m, they can do better (Note: we didn't say m is uniformly random!)
- But they could have done this without obtaining the shares
  - The shares didn't leak any additional information to either party
- Typical crypto goal: **preserving** secrecy
  - View is independent of the message
    - i.e., for all possible values of the message, the view is distributed the same way

$\forall$ view, $\forall$ msg_1, msg_2,
$Pr[view|msg_1] = Pr[view|msg_2]$

Secrecy

Equivalently [Why?]
∀ view, ∀ msg,
(1) Pr[view|msg] = Pr[view]
(2) Pr[msg|view] = Pr[msg] (prior)

- Is the message m really secret?
- Alice or Bob can correctly find the bit m with probability ½, by randomly guessing
  - Worse, if they already know something about m, they can do better (Note: we didn't say m is uniformly random!)
- But they could have done this without obtaining the shares
  - The shares didn't leak any additional information to either party
- Typical crypto goal: **preserving** secrecy
  - View is independent of the message
    - i.e., for all possible values of the message, the view is distributed the same way

∀ view, ∀ msg_1,msg_2,
$Pr[view|msg_1] = Pr[view|msg_2]$

Secrecy

Equivalently [Why?]
∀ view, ∀ msg,
(1) Pr[view|msg] = Pr[view]
(2) Pr[msg|view] = Pr[msg] (prior)

- Is the message m really secret?
- Alice or Bob can correctly find the bit m with probability ½, by randomly guessing
  - Worse, if they already know something about m, they can do better (Note: we didn't say m is uniformly random!)
- But they could have done this without obtaining the shares
  - The shares didn't leak any additional information to either party
- Typical crypto goal: **preserving** secrecy
  - View is independent of the message
    - i.e., for all possible values of the message, the view is distributed the same way

∀ view, ∀ msg_1,msg_2,
Pr[view|msg_1] = Pr[view|msg_2]

Can't get Pr[msg_1|view] = Pr[msg_2|view], unless prior is uniform

Secret-Sharing

Secret-Sharing

- More general secret-sharing

Secret-Sharing

- More general secret-sharing
  - Allow more than two parties (how?)

Secret-Sharing

- More general secret-sharing
  - Allow more than two parties (how?)
  - Privileged subsets of parties should be able to reconstruct the secret (not necessarily just the entire set of parties)

Secret-Sharing

- More general secret-sharing
  - Allow more than two parties (how?)
  - Privileged subsets of parties should be able to reconstruct the secret (not necessarily just the entire set of parties)
- Very useful

Secret-Sharing

- More general secret-sharing
  - Allow more than two parties (how?)
  - Privileged subsets of parties should be able to reconstruct the secret (not necessarily just the entire set of parties)
- Very useful
  - Direct applications (distributed storage of data or keys)

Secret-Sharing

- More general secret-sharing
  - Allow more than two parties (how?)
  - Privileged subsets of parties should be able to reconstruct the secret (not necessarily just the entire set of parties)
- Very useful
  - Direct applications (distributed storage of data or keys)
  - Important component in other cryptographic constructions

Secret-Sharing

- More general secret-sharing
  - Allow more than two parties (how?)
  - Privileged subsets of parties should be able to reconstruct the secret (not necessarily just the entire set of parties)
- Very useful
  - Direct applications (distributed storage of data or keys)
  - Important component in other cryptographic constructions
    - Amplifying secrecy of various primitives

Secret-Sharing

- More general secret-sharing
  - Allow more than two parties (how?)
  - Privileged subsets of parties should be able to reconstruct the secret (not necessarily just the entire set of parties)
- Very useful
  - Direct applications (distributed storage of data or keys)
  - Important component in other cryptographic constructions
    - Amplifying secrecy of various primitives
    - Secure multi-party computation

Secret-Sharing

- More general secret-sharing
  - Allow more than two parties (how?)
  - Privileged subsets of parties should be able to reconstruct the secret (not necessarily just the entire set of parties)
- Very useful
  - Direct applications (distributed storage of data or keys)
  - Important component in other cryptographic constructions
    - Amplifying secrecy of various primitives
    - Secure multi-party computation
    - Attribute-Based Encryption

Secret-Sharing

- More general secret-sharing
  - Allow more than two parties (how?)
  - Privileged subsets of parties should be able to reconstruct the secret (not necessarily just the entire set of parties)
- Very useful
  - Direct applications (distributed storage of data or keys)
  - Important component in other cryptographic constructions
    - Amplifying secrecy of various primitives
    - Secure multi-party computation
    - Attribute-Based Encryption
    - Leakage resilience ...

Threshold Secret-Sharing

Threshold Secret-Sharing

- (n,t)-secret-sharing

Threshold Secret-Sharing

- (n,t)-secret-sharing
  - Divide a message m into n shares $s_1,\ldots,s_n$, such that

Threshold Secret-Sharing

- (n,t)-secret-sharing
  - Divide a message m into n shares $s_1,\ldots,s_n$, such that
    - any t shares are enough to reconstruct the secret

Threshold Secret-Sharing

- (n,t)-secret-sharing
  - Divide a message m into n shares $s_1,\ldots,s_n$, such that
    - any t shares are enough to reconstruct the secret
    - up to t-1 shares should have no information about the secret

Threshold Secret-Sharing

- (n,t)-secret-sharing
  - Divide a message m into n shares $s_1,\ldots,s_n$, such that
    - any t shares are enough to reconstruct the secret
    - up to t-1 shares should have no information about the secret

e.g., $(s_1,\ldots,s_{t-1})$ has the same distribution for every m in the message space

Threshold Secret-Sharing

- (n,t)-secret-sharing
  - Divide a message m into n shares $s_1,\ldots,s_n$, such that
    - any t shares are enough to reconstruct the secret
    - up to t-1 shares should have no information about the secret
- our previous example: (2,2) secret-sharing

e.g., $(s_1,\ldots,s_{t-1})$ has the same distribution for every m in the message space

Threshold Secret-Sharing

Threshold Secret-Sharing

- Construction: (n,n) secret-sharing

Threshold Secret-Sharing

- Construction: (n,n) secret-sharing
  - Message-space = share-space = G, a finite group

Threshold Secret-Sharing

- Construction: (n,n) secret-sharing
  - Message-space = share-space = G, a finite group
    - e.g. G = $\mathbb{Z}_2$ (group of bits, with xor as the group operation)

Threshold Secret-Sharing

- Construction: (n,n) secret-sharing
  - Message-space = share-space = G, a finite group
    - e.g. G = $\mathbb{Z}_2$ (group of bits, with xor as the group operation)
    - or, G = $\mathbb{Z}_2^{\,d}$ (group of d-bit strings)

Threshold Secret-Sharing

- Construction: (n,n) secret-sharing
  - Message-space = share-space = G, a finite group
    - e.g. G = $\mathbb{Z}_2$ (group of bits, with xor as the group operation)
    - or, G = $\mathbb{Z}_2^{\,d}$ (group of d-bit strings)
    - or, G = $\mathbb{Z}_p$ (group of integers mod p)

Threshold Secret-Sharing

- Construction: (n,n) secret-sharing
  - Message-space = share-space = G, a finite group
    - e.g. G = $\mathbb{Z}_2$ (group of bits, with xor as the group operation)
    - or, G = $\mathbb{Z}_2^{\,d}$ (group of d-bit strings)
    - or, G = $\mathbb{Z}_p$ (group of integers mod p)
  - Share(M):

Threshold Secret-Sharing

- Construction: (n,n) secret-sharing
  - Message-space = share-space = G, a finite group
    - e.g. G = $\mathbb{Z}_2$ (group of bits, with xor as the group operation)
    - or, G = $\mathbb{Z}_2^{\,d}$ (group of d-bit strings)
    - or, G = $\mathbb{Z}_p$ (group of integers mod p)
  - Share(M):
    - Pick $s_1,\ldots,s_{n-1}$ uniformly at random from G

Threshold Secret-Sharing

- Construction: (n,n) secret-sharing
  - Message-space = share-space = G, a finite group
    - e.g. G = $\mathbb{Z}_2$ (group of bits, with xor as the group operation)
    - or, G = $\mathbb{Z}_2^{\,d}$ (group of d-bit strings)
    - or, G = $\mathbb{Z}_p$ (group of integers mod p)
  - Share(M):
    - Pick $s_1,\ldots,s_{n-1}$ uniformly at random from G
    - Let $s_n = -(s_1 + \ldots + s_{n-1}) + M$

Threshold Secret-Sharing

- Construction: (n,n) secret-sharing
  - Message-space = share-space = G, a finite group
    - e.g. G = $\mathbb{Z}_2$ (group of bits, with xor as the group operation)
    - or, G = $\mathbb{Z}_2^{\,d}$ (group of d-bit strings)
    - or, G = $\mathbb{Z}_p$ (group of integers mod p)
  - Share(M):
    - Pick $s_1,\ldots,s_{n-1}$ uniformly at random from G
    - Let $s_n = -(s_1 + \ldots + s_{n-1}) + M$
  - Reconstruct($s_1,\ldots,s_n$): $M = s_1 + \ldots + s_n$

Threshold Secret-Sharing

- Construction: (n,n) secret-sharing
  - Message-space = share-space = G, a finite group
    - e.g. G = $\mathbb{Z}_2$ (group of bits, with xor as the group operation)
    - or, G = $\mathbb{Z}_2^{\,d}$ (group of d-bit strings)
    - or, G = $\mathbb{Z}_p$ (group of integers mod p)
  - Share(M):
    - Pick $s_1,\ldots,s_{n-1}$ uniformly at random from G
    - Let $s_n = -(s_1 + \ldots + s_{n-1}) + M$
  - Reconstruct($s_1,\ldots,s_n$): $M = s_1 + \ldots + s_n$
  - Claim: This is an (n,n) secret-sharing scheme [Why?]

Threshold Secret-Sharing

Additive Secret-Sharing

- Construction: (n,n) secret-sharing
  - Message-space = share-space = G, a finite group
    - e.g. G = $\mathbb{Z}_2$ (group of bits, with xor as the group operation)
    - or, G = $\mathbb{Z}_2^{\,d}$ (group of d-bit strings)
    - or, G = $\mathbb{Z}_p$ (group of integers mod p)
  - Share(M):
    - Pick $s_1,\ldots,s_{n-1}$ uniformly at random from G
    - Let $s_n = -(s_1 + \ldots + s_{n-1}) + M$
  - Reconstruct($s_1,\ldots,s_n$): $M = s_1 + \ldots + s_n$
  - Claim: This is an (n,n) secret-sharing scheme [Why?]

PROOF

Additive Secret-Sharing: Proof

- Share(M):
  - Pick $s_1,\ldots,s_{n-1}$ uniformly at random from G
  - Let $s_n = M - (s_1 + \ldots + s_{n-1})$
- Reconstruct($s_1,\ldots,s_n$): $M = s_1 + \ldots + s_n$
- Claim: Upto n-1 shares give no information about M
- Proof: Let $T \subseteq \{1,\ldots,n\}$, $|T| = n-1$. We shall show that $\{ s_i \}_{i\in T}$ is distributed the same way (in fact, uniformly) irrespective of what M is.
  - For concreteness consider $T = \{2,\ldots,n\}$. Fix any (n-1)-tuple of elements in G, $(g_1,\ldots,g_{n-1}) \in G^{n-1}$. To prove $Pr[(s_2,\ldots,s_n)=(g_1,\ldots,g_{n-1})]$ is same for all M.
  - Fix any M.
  - $(s_2,\ldots,s_n) = (g_1,\ldots,g_{n-1}) \Leftrightarrow (s_2,\ldots,s_{n-1}) = (g_1,\ldots,g_{n-2})$ and $s_1 = M-(g_1+\ldots+g_{n-1})$.
  - So $Pr[(s_2,\ldots,s_n)=(g_1,\ldots,g_{n-1})] = Pr[(s_1,\ldots,s_{n-1})=(a,g_1,\ldots,g_{n-2})]$, $a:=(M-(g_1+\ldots+g_{n-1})$
  - But $Pr[(s_1,\ldots,s_{n-1})=(a,g_1,\ldots,g_{n-2})] = 1/|G|^{n-1}$, since $(s_1,\ldots,s_{n-1})$ are picked uniformly at random from G
  - Hence $Pr[(s_2,\ldots,s_n)=(g_1,\ldots,g_{n-1})] = 1/|G|^{n-1}$, irrespective of M. □

Threshold Secret-Sharing

Threshold Secret-Sharing

- Construction: (n,2) secret-sharing

Threshold Secret-Sharing

- Construction: (n,2) secret-sharing
- Message-space = share-space = F, a field (e.g. integers mod a prime, $\mathbb{F}_p$)

Threshold Secret-Sharing

- Construction: (n,2) secret-sharing
- Message-space = share-space = F, a field (e.g. integers mod a prime, $\mathbb{F}_p$)
  - Share(M): pick random r. Let $s_i = r \cdot a_i + M$ (for $i=1,\ldots,n < |F|$)

Threshold Secret-Sharing

- Construction: (n,2) secret-sharing
- Message-space = share-space = F, a field (e.g. integers mod a prime, $\mathbb{F}_p$)
  - Share(M): pick random r. Let $s_i = r \cdot a_i + M$ (for $i=1,\ldots,n < |F|$)

a_i are n distinct, non-zero field elements

Threshold Secret-Sharing

- Construction: (n,2) secret-sharing
- Message-space = share-space = F, a field (e.g. integers mod a prime, $\mathbb{F}_p$)
  - Share(M): pick random r. Let $s_i = r \cdot a_i + M$ (for $i=1,\ldots,n < |F|$)
  - Reconstruct(s_i, s_j): $r = (s_i - s_j)/(a_i - a_j)$; $M = s_i - r \cdot a_i$

a_i are n distinct, non-zero field elements

Threshold Secret-Sharing

- Construction: (n,2) secret-sharing
- Message-space = share-space = F, a field (e.g. integers mod a prime, $\mathbb{F}_p$)
  - Share(M): pick random r. Let $s_i = r \cdot a_i + M$ (for $i=1,\ldots,n < |F|$)
  - Reconstruct(s_i, s_j): $r = (s_i - s_j)/(a_i - a_j)$; $M = s_i - r \cdot a_i$
  - Each s_i by itself is uniformly distributed, irrespective of M [Why?]

a_i are n distinct, non-zero field elements

Threshold Secret-Sharing

- Construction: (n,2) secret-sharing
- Message-space = share-space = F, a field (e.g. integers mod a prime, $\mathbb{F}_p$)
  - Share(M): pick random r. Let $s_i = r \cdot a_i + M$ (for $i=1,\ldots,n < |F|$)
  - Reconstruct(s_i, s_j): $r = (s_i - s_j)/(a_i - a_j)$; $M = s_i - r \cdot a_i$
  - Each s_i by itself is uniformly distributed, irrespective of M [Why?]

a_i are n distinct, non-zero field elements

Since a_i^{-1} exists, exactly one solution for $r \cdot a_i + M = d$, for every value of d

Threshold Secret-Sharing

- Construction: (n,2) secret-sharing
- Message-space = share-space = F, a field (e.g. integers mod a prime, $\mathbb{F}_p$)
  - Share(M): pick random r. Let $s_i = r \cdot a_i + M$ (for $i=1,\ldots,n < |F|$)
  - Reconstruct(s_i, s_j): $r = (s_i - s_j)/(a_i - a_j)$; $M = s_i - r \cdot a_i$
  - Each s_i by itself is uniformly distributed, irrespective of M [Why?]
  - "Geometric" interpretation

a_i are n distinct, non-zero field elements

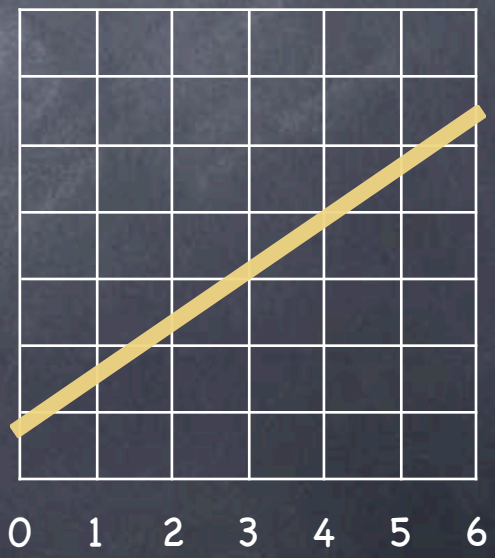
Since a_i^{-1} exists, exactly one solution for $r \cdot a_i + M = d$, for every value of d

Threshold Secret-Sharing

- Construction: (n,2) secret-sharing
- Message-space = share-space = F, a field (e.g. integers mod a prime, $\mathbb{F}_p$)
  - Share(M): pick random r. Let $s_i = r \cdot a_i + M$ (for $i=1,\ldots,n < |F|$)
  - Reconstruct(s_i, s_j): $r = (s_i - s_j)/(a_i - a_j)$; $M = s_i - r \cdot a_i$
  - Each s_i by itself is uniformly distributed, irrespective of M [Why?]
  - "Geometric" interpretation
    - Sharing picks a random "line" y = f(x), such that f(0)=M. Shares $s_i = f(a_i)$.

a_i are n distinct, non-zero field elements

Since a_i^{-1} exists, exactly one solution for $r \cdot a_i + M = d$, for every value of d

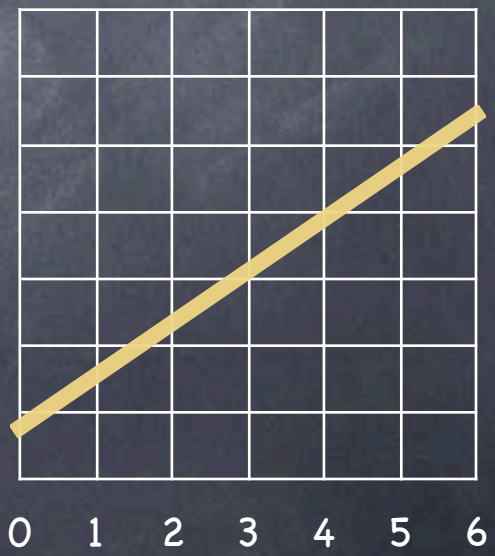


Threshold Secret-Sharing

- Construction: (n,2) secret-sharing
- Message-space = share-space = F, a field (e.g. integers mod a prime, $\mathbb{F}_p$)
  - Share(M): pick random r. Let $s_i = r \cdot a_i + M$ (for $i=1,\ldots,n < |F|$)
  - Reconstruct(s_i, s_j): $r = (s_i - s_j)/(a_i - a_j)$; $M = s_i - r \cdot a_i$
  - Each s_i by itself is uniformly distributed, irrespective of M [Why?]
  - "Geometric" interpretation
    - Sharing picks a random "line" y = f(x), such that f(0)=M. Shares $s_i = f(a_i)$.
    - s_i is independent of M: exactly one line passing through (a_i,s_i) and (0,M') for any secret M'

a_i are n distinct, non-zero field elements

Since a_i^{-1} exists, exactly one solution for $r \cdot a_i + M = d$, for every value of d

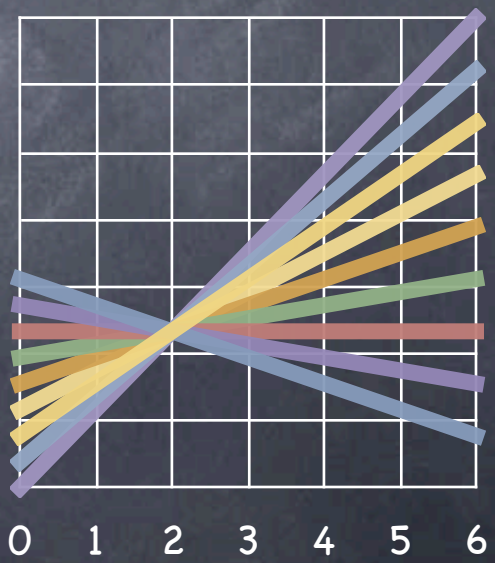


Threshold Secret-Sharing

- Construction: (n,2) secret-sharing
- Message-space = share-space = F, a field (e.g. integers mod a prime, $\mathbb{F}_p$)
  - Share(M): pick random r. Let $s_i = r \cdot a_i + M$ (for $i=1,\ldots,n < |F|$)
  - Reconstruct(s_i, s_j): $r = (s_i - s_j)/(a_i - a_j)$; $M = s_i - r \cdot a_i$
  - Each s_i by itself is uniformly distributed, irrespective of M [Why?]
  - "Geometric" interpretation
    - Sharing picks a random "line" y = f(x), such that f(0)=M. Shares $s_i = f(a_i)$.
    - s_i is independent of M: exactly one line passing through (a_i,s_i) and (0,M') for any secret M'

a_i are n distinct, non-zero field elements

Since a_i^{-1} exists, exactly one solution for $r \cdot a_i + M = d$, for every value of d

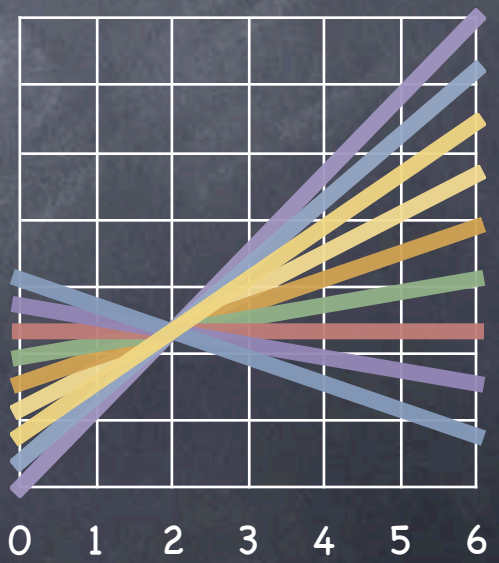


Threshold Secret-Sharing

- Construction: (n,2) secret-sharing
- Message-space = share-space = F, a field (e.g. integers mod a prime, $\mathbb{F}_p$)
  - Share(M): pick random r. Let $s_i = r \cdot a_i + M$ (for $i=1,\ldots,n < |F|$)
  - Reconstruct(s_i, s_j): $r = (s_i - s_j)/(a_i - a_j)$; $M = s_i - r \cdot a_i$
  - Each s_i by itself is uniformly distributed, irrespective of M [Why?]
  - "Geometric" interpretation
    - Sharing picks a random "line" y = f(x), such that f(0)=M. Shares $s_i = f(a_i)$.
    - s_i is independent of M: exactly one line passing through (a_i,s_i) and (0,M') for any secret M'
    - But can reconstruct the line from two points!

a_i are n distinct, non-zero field elements

Since a_i^{-1} exists, exactly one solution for $r \cdot a_i + M = d$, for every value of d



PROOF

(n,2) Secret-Sharing: Proof

- Share(M): pick random $r \leftarrow F$. Let $s_i = r \cdot a_i + M$ (for $i=1,\ldots,n < |F|$)
- Reconstruct(s_i, s_j): $r = (s_i - s_j)/(a_i - a_j)$; $M = s_i - r \cdot a_i$

- Claim: Any one share gives no information about M
- Proof: For any $i \in \{1,..,n\}$ we shall show that s_i is distributed the same way (in fact, uniformly) irrespective of what M is.
- Consider any $g \in F$. We shall show that Pr[$s_i=g$] is independent of M.
- Fix any M.
- For any $g \in F$, $s_i = g \Leftrightarrow r \cdot i + M = g \Leftrightarrow r = (g-M) \cdot a_i^{-1}$ (since $a_i \neq 0$)
- So, Pr[$s_i=g$] = Pr[$r=(g-M) \cdot a_i^{-1}$] = $1/|F|$, since r is chosen uniformly at random

□

Threshold Secret-Sharing

Threshold Secret-Sharing

- (n,t) secret-sharing in a field F

Threshold Secret-Sharing

- (n,t) secret-sharing in a field F
- Generalizing the geometric/algebraic view: instead of lines, use polynomials

Threshold Secret-Sharing

Shamir Secret-Sharing

- (n,t) secret-sharing in a field F
- Generalizing the geometric/algebraic view: instead of lines, use polynomials

Threshold Secret-Sharing

Shamir Secret-Sharing

- (n,t) secret-sharing in a field F
- Generalizing the geometric/algebraic view: instead of lines, use polynomials
  - Share(m): Pick a random degree t-1 polynomial f(X), such that f(0)=M. Shares are $s_i = f(a_i)$.

Threshold Secret-Sharing

Shamir Secret-Sharing

- (n,t) secret-sharing in a field F
- Generalizing the geometric/algebraic view: instead of lines, use polynomials
  - Share(m): Pick a random degree t-1 polynomial f(X), such that f(0)=M. Shares are $s_i = f(a_i)$.
    - Random polynomial with f(0)=M: $c_0 + c_1X + c_2X^2 + \ldots + c_{t-1}X^{t-1}$ by picking c_0=M and $c_1,\ldots,c_{t-1}$ at random.

Threshold Secret-Sharing

Shamir Secret-Sharing

- (n,t) secret-sharing in a field F
- Generalizing the geometric/algebraic view: instead of lines, use polynomials
  - Share(m): Pick a random degree t-1 polynomial f(X), such that f(0)=M. Shares are $s_i = f(a_i)$.
    - Random polynomial with f(0)=M: $c_0 + c_1X + c_2X^2 + \ldots + c_{t-1}X^{t-1}$ by picking c_0=M and $c_1,\ldots,c_{t-1}$ at random.
  - Reconstruct($s_1,\ldots,s_t$): Lagrange interpolation to find M=c_0

Threshold Secret-Sharing

Shamir Secret-Sharing

- (n,t) secret-sharing in a field F
- Generalizing the geometric/algebraic view: instead of lines, use polynomials
  - Share(m): Pick a random degree t-1 polynomial f(X), such that f(0)=M. Shares are $s_i = f(a_i)$.
    - Random polynomial with f(0)=M: $c_0 + c_1X + c_2X^2 + \ldots + c_{t-1}X^{t-1}$ by picking c_0=M and $c_1,\ldots,c_{t-1}$ at random.
  - Reconstruct($s_1,\ldots,s_t$): Lagrange interpolation to find M=c_0
    - Need t points to reconstruct the polynomial. Given t-1 points, out of $|F|^{t-1}$ polynomials passing through (0,M') (for any M') there is exactly one that passes through the t-1 points

Lagrange Interpolation

Lagrange Interpolation

- Given t distinct points on a degree t-1 polynomial (univariate, over some field of more than t elements), reconstruct the entire polynomial (i.e., find all t co-efficients)

Lagrange Interpolation

- Given t distinct points on a degree t-1 polynomial (univariate, over some field of more than t elements), reconstruct the entire polynomial (i.e., find all t co-efficients)
  - t variables: $c_0,\ldots,c_{t-1}$. t equations: $1.c_0 + a_i.c_1 + a_i^2.c_2 + \ldots a_i^{t-1}.c_{t-1} = s_i$

Lagrange Interpolation

- Given t distinct points on a degree t-1 polynomial (univariate, over some field of more than t elements), reconstruct the entire polynomial (i.e., find all t co-efficients)
  - t variables: $c_0,\ldots,c_{t-1}$. t equations: $1.c_0 + a_i.c_1 + a_i^2.c_2 + \ldots a_i^{t-1}.c_{t-1} = s_i$
  - A linear system: $W\mathbf{c}=\mathbf{s}$, where W is a txt matrix with i^{th} row, $W_i = (1\ a_i\ a_i^2 \ldots a_i^{t-1})$

Lagrange Interpolation

- Given t distinct points on a degree t-1 polynomial (univariate, over some field of more than t elements), reconstruct the entire polynomial (i.e., find all t co-efficients)
  - t variables: $c_0,\ldots,c_{t-1}$. t equations: $1.c_0 + a_i.c_1 + a_i^2.c_2 + \ldots a_i^{t-1}.c_{t-1} = s_i$
  - A linear system: $W\mathbf{c}=\mathbf{s}$, where W is a txt matrix with i^{th} row, $W_i = (1\ a_i\ a_i^2 \ldots a_i^{t-1})$
  - W (called the Vandermonde matrix) is invertible

Lagrange Interpolation

- Given t distinct points on a degree t-1 polynomial (univariate, over some field of more than t elements), reconstruct the entire polynomial (i.e., find all t co-efficients)
  - t variables: $c_0,\ldots,c_{t-1}$. t equations: $1.c_0 + a_i.c_1 + a_i^2.c_2 + \ldots a_i^{t-1}.c_{t-1} = s_i$
  - A linear system: $W\mathbf{c}=\mathbf{s}$, where W is a txt matrix with i^{th} row, $W_i = (1\ a_i\ a_i^2 \ldots a_i^{t-1})$
  - W (called the Vandermonde matrix) is invertible
    - $\mathbf{c} = W^{-1}\mathbf{s}$

More General Access Structures

More General Access Structures

- (n,t)-secret-sharing allowed any t (or more) parties to reconstruct the secret

More General Access Structures

- (n,t)-secret-sharing allowed any t (or more) parties to reconstruct the secret
  - i.e., "access structure" $\mathcal{A}$ = {S: |S| ≥ t }, is the set of all subsets of parties who can reconstruct the secret

More General Access Structures

- (n,t)-secret-sharing allowed any t (or more) parties to reconstruct the secret
  - i.e., "access structure" $\mathcal{A}$ = {S: |S| ≥ t }, is the set of all subsets of parties who can reconstruct the secret
  - In general access structure could be any monotonic set of subsets

More General Access Structures

- (n,t)-secret-sharing allowed any t (or more) parties to reconstruct the secret
  - i.e., "access structure" $\mathcal{A}$ = {S: |S| ≥ t }, is the set of all subsets of parties who can reconstruct the secret
  - In general access structure could be any monotonic set of subsets

If $S^* \in \mathcal{A}$, then for all $S \supseteq S^*$, $S \in \mathcal{A}$.

More General Access Structures

- (n,t)-secret-sharing allowed any t (or more) parties to reconstruct the secret
  - i.e., "access structure" $\mathcal{A}$ = {S: |S| ≥ t }, is the set of all subsets of parties who can reconstruct the secret
  - In general access structure could be any monotonic set of subsets

If $S^* \in \mathcal{A}$, then for all $S \supseteq S^*$, $S \in \mathcal{A}$.

- Shamir's secret-sharing solves threshold secret-sharing. How about the others?

More General Access Structures

More General Access Structures

- Idea: For arbitrary monotonic access structure $\mathcal{A}$, there is a "basis" $\mathcal{B}$ of minimal sets in $\mathcal{A}$. For each S in $\mathcal{B}$ generate an (|S|,|S|) sharing, and distribute them to the members of S.

More General Access Structures

- Idea: For arbitrary monotonic access structure $\mathcal{A}$, there is a "basis" $\mathcal{B}$ of minimal sets in $\mathcal{A}$. For each S in $\mathcal{B}$ generate an (|S|,|S|) sharing, and distribute them to the members of S.
  - Works, but very "inefficient"

More General Access Structures

- Idea: For arbitrary monotonic access structure $\mathcal{A}$, there is a "basis" $\mathcal{B}$ of minimal sets in $\mathcal{A}$. For each S in $\mathcal{B}$ generate an (|S|,|S|) sharing, and distribute them to the members of S.
  - Works, but very "inefficient"
    - How big is $\mathcal{B}$? (Say when $\mathcal{A}$ is a threshold access structure)

More General Access Structures

- Idea: For arbitrary monotonic access structure $\mathcal{A}$, there is a "basis" $\mathcal{B}$ of minimal sets in $\mathcal{A}$. For each S in $\mathcal{B}$ generate an (|S|,|S|) sharing, and distribute them to the members of S.
  - Works, but very "inefficient"
    - How big is $\mathcal{B}$? (Say when $\mathcal{A}$ is a threshold access structure)

$|\mathcal{B}|$ = (n choose t)

More General Access Structures

- Idea: For arbitrary monotonic access structure $\mathcal{A}$, there is a "basis" $\mathcal{B}$ of minimal sets in $\mathcal{A}$. For each S in $\mathcal{B}$ generate an (|S|,|S|) sharing, and distribute them to the members of S.
  - Works, but very "inefficient"
    - How big is $\mathcal{B}$? (Say when $\mathcal{A}$ is a threshold access structure)
    - Total share complexity = $\Sigma_{S \in \mathcal{B}} |S|$ field elements. (Compare with Shamir's scheme: n field elements in all.)

$|\mathcal{B}|$ = (n choose t)

More General Access Structures

- Idea: For arbitrary monotonic access structure $\mathcal{A}$, there is a "basis" $\mathcal{B}$ of minimal sets in $\mathcal{A}$. For each S in $\mathcal{B}$ generate an (|S|,|S|) sharing, and distribute them to the members of S.
  - Works, but very "inefficient"
    - How big is $\mathcal{B}$? (Say when $\mathcal{A}$ is a threshold access structure)
    - Total share complexity = $\Sigma_{S \in \mathcal{B}} |S|$ field elements. (Compare with Shamir's scheme: n field elements in all.)

$|\mathcal{B}|$ = (n choose t)

t·(n choose t)

More General Access Structures

- Idea: For arbitrary monotonic access structure $\mathcal{A}$, there is a "basis" $\mathcal{B}$ of minimal sets in $\mathcal{A}$. For each S in $\mathcal{B}$ generate an (|S|,|S|) sharing, and distribute them to the members of S.
  - Works, but very "inefficient"
    - How big is $\mathcal{B}$? (Say when $\mathcal{A}$ is a threshold access structure)
    - Total share complexity = $\Sigma_{S \in \mathcal{B}} |S|$ field elements. (Compare with Shamir's scheme: n field elements in all.)
  - More efficient schemes known for large classes of access structures

$|\mathcal{B}|$ = (n choose t)

t·(n choose t)

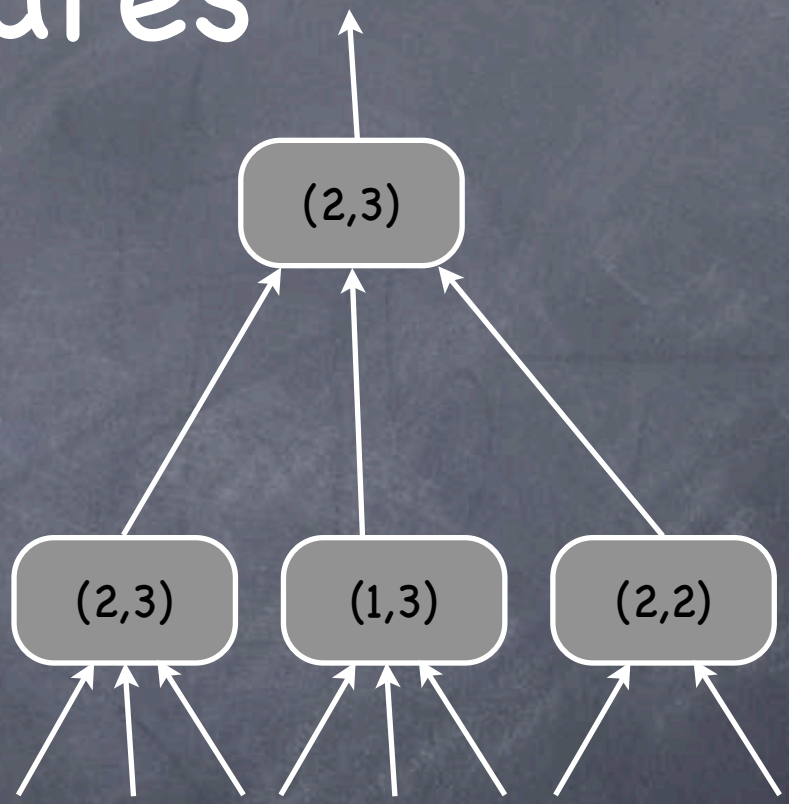
More General Access Structures

More General Access Structures

- A simple generalization of threshold access structures

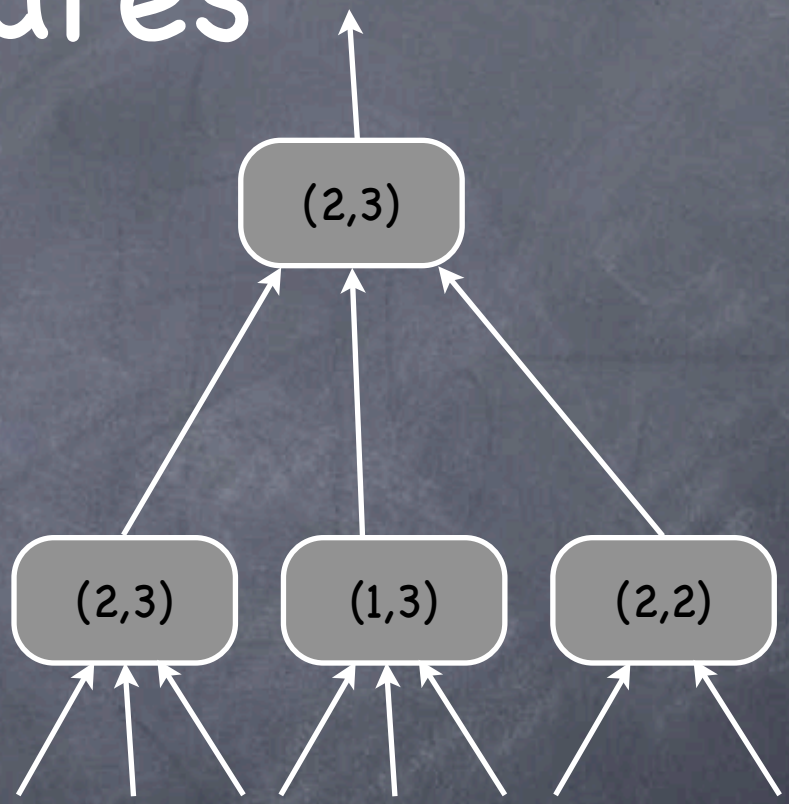
More General Access Structures

- A simple generalization of threshold access structures
  - A threshold tree to specify the access structure



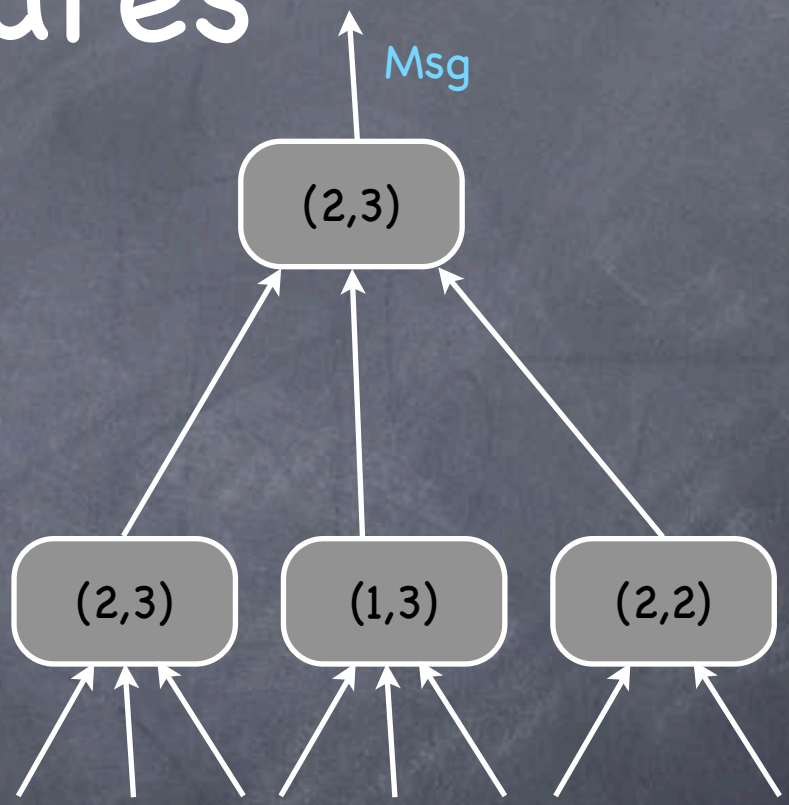
More General Access Structures

- A simple generalization of threshold access structures
  - A threshold tree to specify the access structure
  - Can realize by recursively threshold secret-sharing the shares



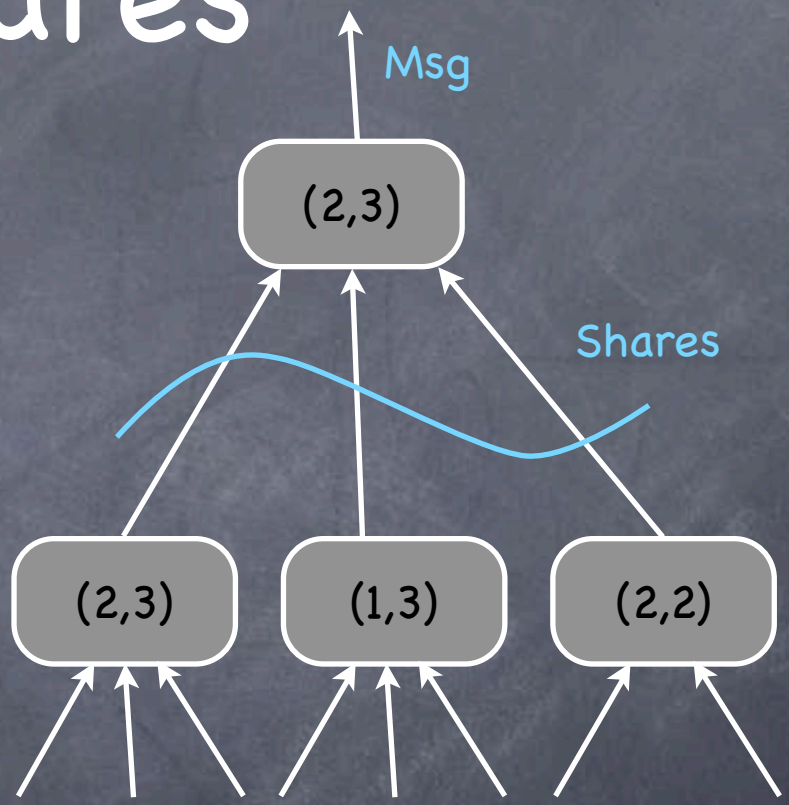
More General Access Structures

- A simple generalization of threshold access structures
  - A threshold tree to specify the access structure
  - Can realize by recursively threshold secret-sharing the shares



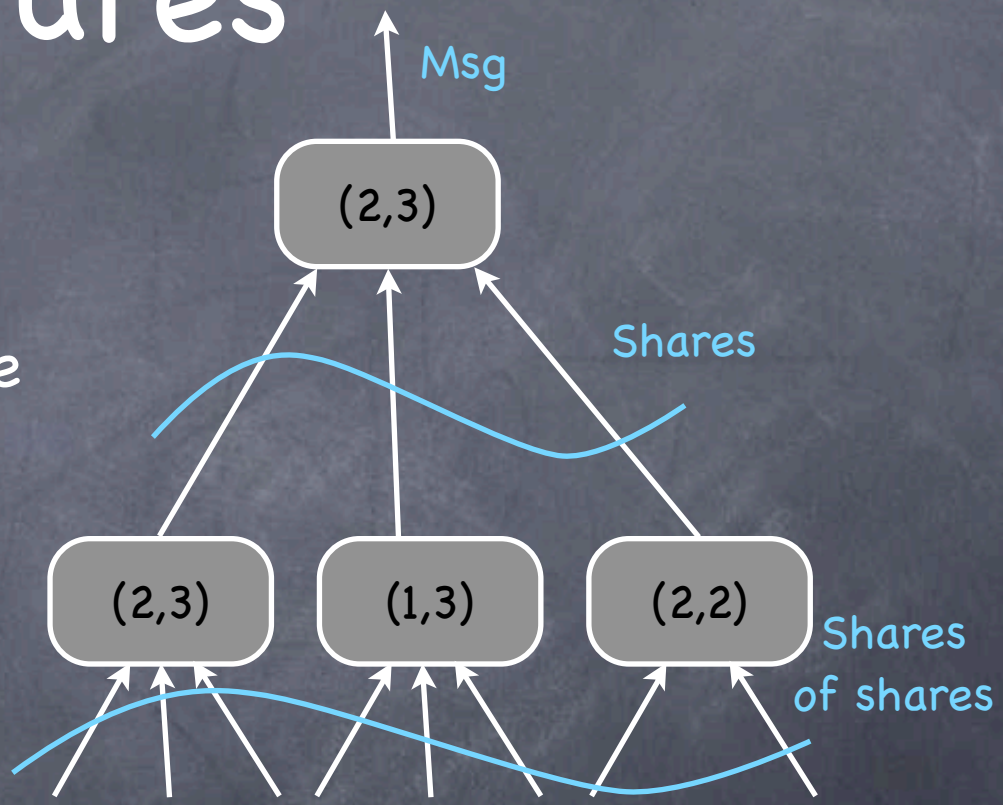
More General Access Structures

- A simple generalization of threshold access structures
  - A threshold tree to specify the access structure
  - Can realize by recursively threshold secret-sharing the shares



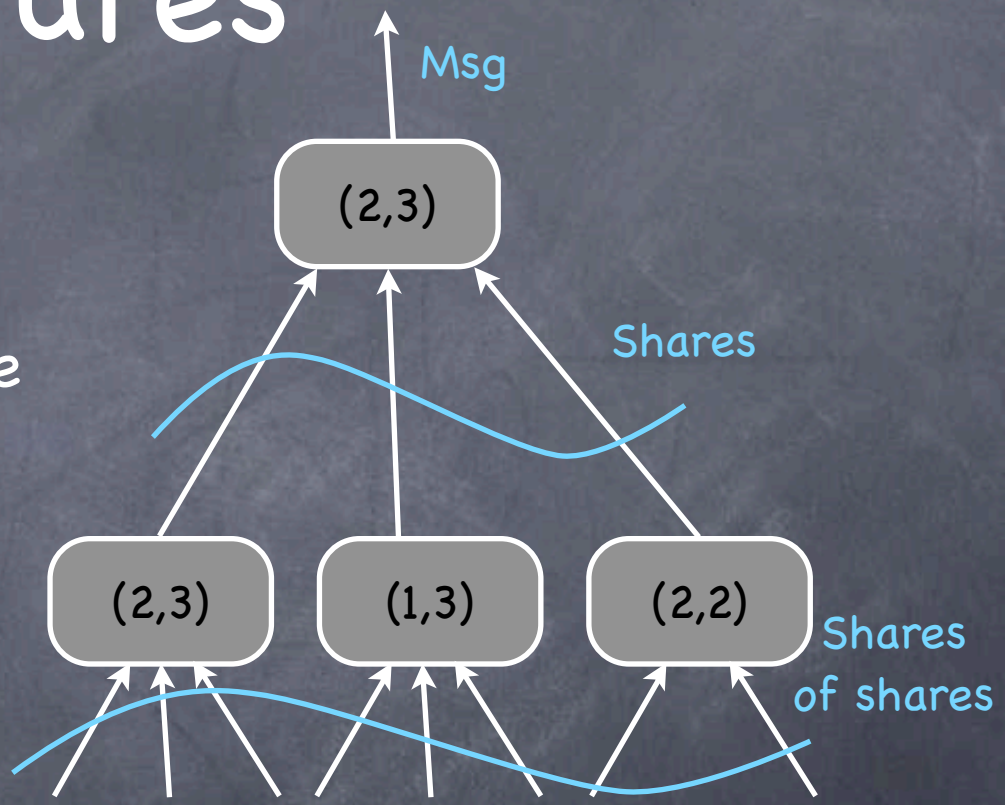
More General Access Structures

- A simple generalization of threshold access structures
  - A threshold tree to specify the access structure
  - Can realize by recursively threshold secret-sharing the shares



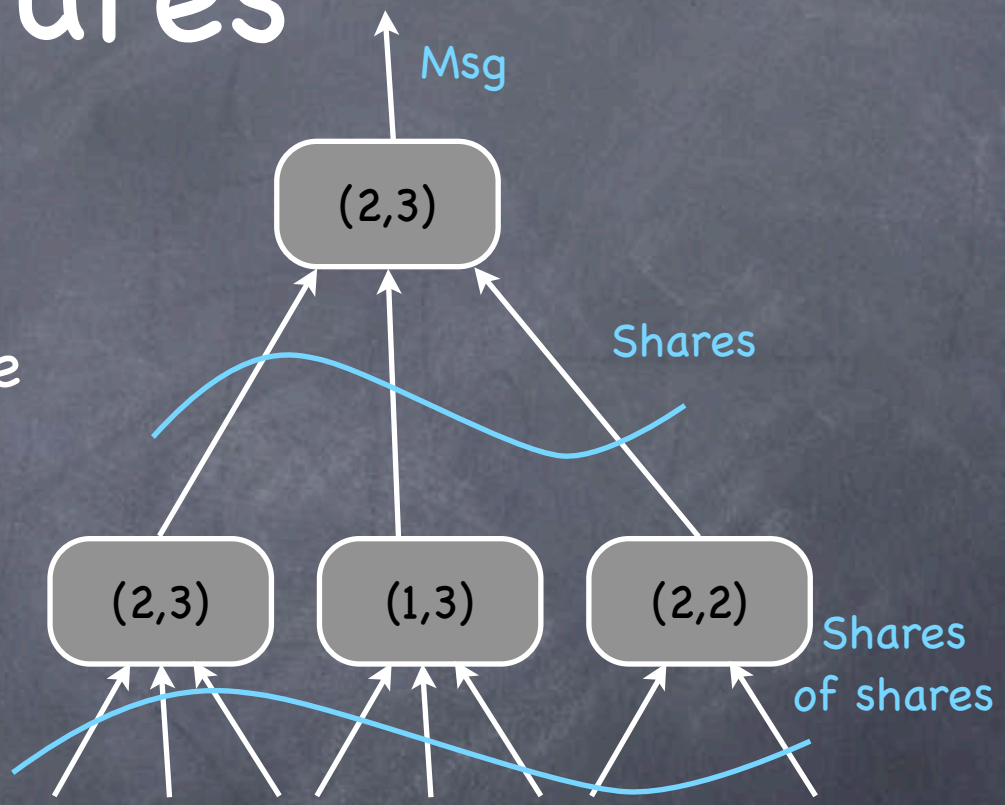
More General Access Structures

- A simple generalization of threshold access structures
  - A threshold tree to specify the access structure
  - Can realize by recursively threshold secret-sharing the shares
- A special case of access structures that can be specified using "monotone span programs"



More General Access Structures

- A simple generalization of threshold access structures
  - A threshold tree to specify the access structure
  - Can realize by recursively threshold secret-sharing the shares
- A special case of access structures that can be specified using "monotone span programs"
  - Admits linear secret-sharing



Linear Secret-Sharing

Linear Secret-Sharing

- Share(M): For some fixed $n \times t$ matrix W, let $\overline{s} = W \cdot \overline{c}$, where $c_0 = M$ and other $t-1$ coordinates are random.

Linear Secret-Sharing

- Share(M): For some fixed $n \times t$ matrix W, let $\overline{s} = W \cdot \overline{c}$, where $c_0 = M$ and other $t-1$ coordinates are random.
  - The shares are subsets of coordinates of $\overline{s}$

Linear Secret-Sharing

- Share(M): For some fixed $n \times t$ matrix W, let $\overline{s} = W \cdot \overline{c}$, where $c_0 = M$ and other $t-1$ coordinates are random.
  - The shares are subsets of coordinates of $\overline{s}$

Shamir Secret-Sharing is of this form

Linear Secret-Sharing

- Share(M): For some fixed $n \times t$ matrix W, let $\overline{s} = W \cdot \overline{c}$, where $c_0 = M$ and other $t-1$ coordinates are random.
  - The shares are subsets of coordinates of $\overline{s}$

Shamir Secret-Sharing is of this form

- Reconstruction: pool together all the available coordinates of $\overline{s}$; can reconstruct if there are enough equations to solve for c_0

Linear Secret-Sharing

- Share(M): For some fixed n×t matrix W, let $\overline{s} = W \cdot \overline{c}$, where $c_0 = M$ and other t-1 coordinates are random.
  - The shares are subsets of coordinates of $\overline{s}$

Shamir Secret-Sharing is of this form

- Reconstruction: pool together all the available coordinates of $\overline{s}$; can reconstruct if there are enough equations to solve for c_0
  - If not reconstructible, shares independent of secret

Linear Secret-Sharing

- Share(M): For some fixed $n \times t$ matrix W, let $\overline{s} = W \cdot \overline{c}$, where $c_0 = M$ and other $t-1$ coordinates are random.
  - The shares are subsets of coordinates of $\overline{s}$

Shamir Secret-Sharing is of this form

- Reconstruction: pool together all the available coordinates of $\overline{s}$; can reconstruct if there are enough equations to solve for c_0
  - If not reconstructible, shares independent of secret
- May not correspond to a threshold access structure

Linear Secret-Sharing

- Share(M): For some fixed $n \times t$ matrix W, let $\overline{s} = W \cdot \overline{c}$, where $c_0 = M$ and other t-1 coordinates are random.
  - The shares are subsets of coordinates of $\overline{s}$

Shamir Secret-Sharing is of this form

- Reconstruction: pool together all the available coordinates of $\overline{s}$; can reconstruct if there are enough equations to solve for c_0
  - If not reconstructible, shares independent of secret
- May not correspond to a threshold access structure
- Reconstruction too is a linear combination of available shares (coefficients depending on which subset of shares available)

Linear Secret-Sharing

Linear Secret-Sharing

- Linearity of linear secret-sharing:

Linear Secret-Sharing

- Linearity of linear secret-sharing:
  - If two secrets m_1, $m_2 \in F$ have been shared and parties get shares $\{x_i\}$ and $\{y_i\}$ (also elements of F) as shares, then each party can locally obtain sharing $\{z_i\}$ of am_1+bm_2

Linear Secret-Sharing

- Linearity of linear secret-sharing:
  - If two secrets m_1, $m_2 \in F$ have been shared and parties get shares $\{x_i\}$ and $\{y_i\}$ (also elements of F) as shares, then each party can locally obtain sharing $\{z_i\}$ of am_1+bm_2
    - $z_i = ax_i + by_i$

Linear Secret-Sharing

- Linearity of linear secret-sharing:
  - If two secrets $m_1, m_2 \in F$ have been shared and parties get shares $\{x_i\}$ and $\{y_i\}$ (also elements of F) as shares, then each party can locally obtain sharing $\{z_i\}$ of am_1+bm_2
    - $z_i = ax_i + by_i$

$$\overline{x} = W \cdot \overline{c}_1$$
$$\overline{y} = W \cdot \overline{c}_2$$
$$\overline{z} = W \cdot (a\overline{c}_1 + b\overline{c}_2)$$

Linear Secret-Sharing

- Linearity of linear secret-sharing:
  - If two secrets $m_1, m_2 \in F$ have been shared and parties get shares $\{x_i\}$ and $\{y_i\}$ (also elements of F) as shares, then each party can locally obtain sharing $\{z_i\}$ of am_1+bm_2
    - $z_i = ax_i + by_i$
  - Useful in secure multiparty computation (later)

$$\overline{x} = W \cdot \overline{c}_1$$
$$\overline{y} = W \cdot \overline{c}_2$$
$$\overline{z} = W \cdot (a\overline{c}_1 + b\overline{c}_2)$$

Linear Secret-Sharing

- Linearity of linear secret-sharing:
  - If two secrets $m_1, m_2 \in F$ have been shared and parties get shares $\{x_i\}$ and $\{y_i\}$ (also elements of F) as shares, then each party can locally obtain sharing $\{z_i\}$ of am_1+bm_2
    - $z_i = ax_i + by_i$
  - Useful in secure multiparty computation (later)

$$\overline{x} = W \cdot \overline{c}_1$$
$$\overline{y} = W \cdot \overline{c}_2$$
$$\overline{z} = W \cdot (a\overline{c}_1 + b\overline{c}_2)$$

- Simple(st) example: from additive shares for two bits m_1 and m_2, n parties can locally obtain an additive sharing of $m_1 \oplus m_2$

Linear Secret-Sharing

- Linearity of linear secret-sharing:
  - If two secrets $m_1, m_2 \in F$ have been shared and parties get shares $\{x_i\}$ and $\{y_i\}$ (also elements of F) as shares, then each party can locally obtain sharing $\{z_i\}$ of am_1+bm_2
    - $z_i = ax_i + by_i$
  - Useful in secure multiparty computation (later)

$$\overline{x} = W \cdot \overline{c}_1$$
$$\overline{y} = W \cdot \overline{c}_2$$
$$\overline{z} = W \cdot (a\overline{c}_1 + b\overline{c}_2)$$

- Simple(st) example: from additive shares for two bits m_1 and m_2, n parties can locally obtain an additive sharing of $m_1 \oplus m_2$
  - Gives a "private summation" protocol

Linear Secret-Sharing

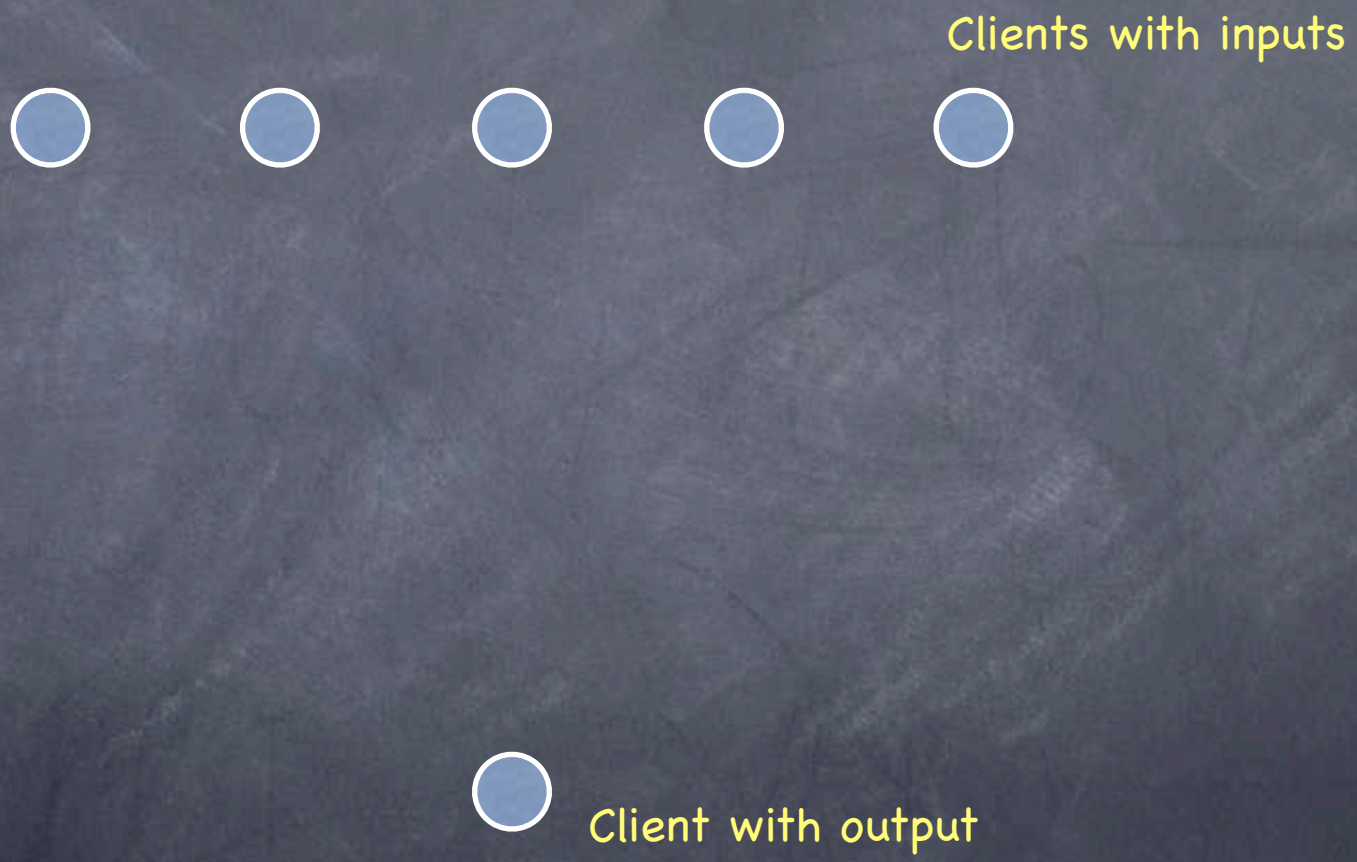
- Gives a "private summation" protocol

Linear Secret-Sharing

- Gives a "private summation" protocol

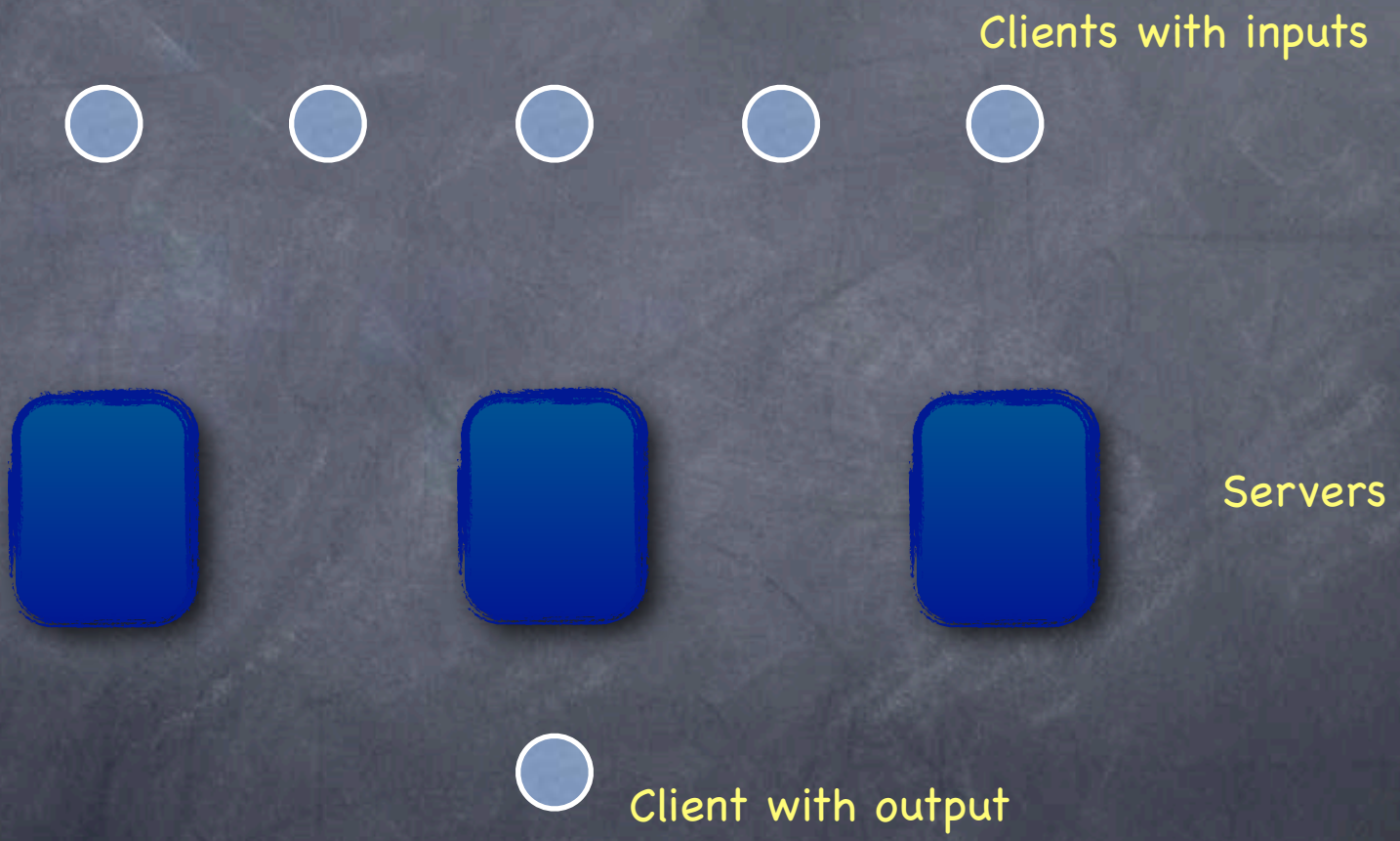
Linear Secret-Sharing

- Gives a "private summation" protocol



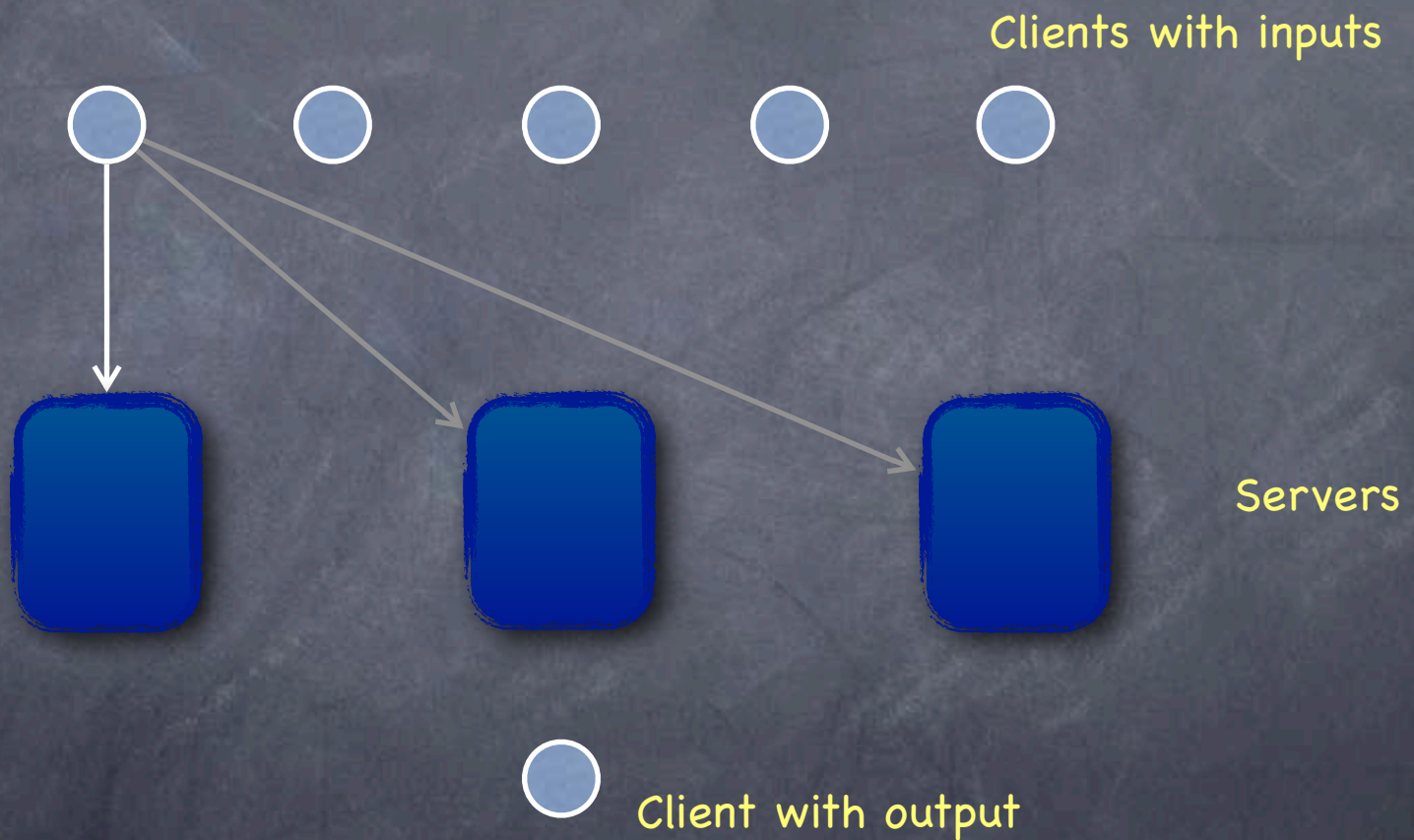
Linear Secret-Sharing

- Gives a "private summation" protocol



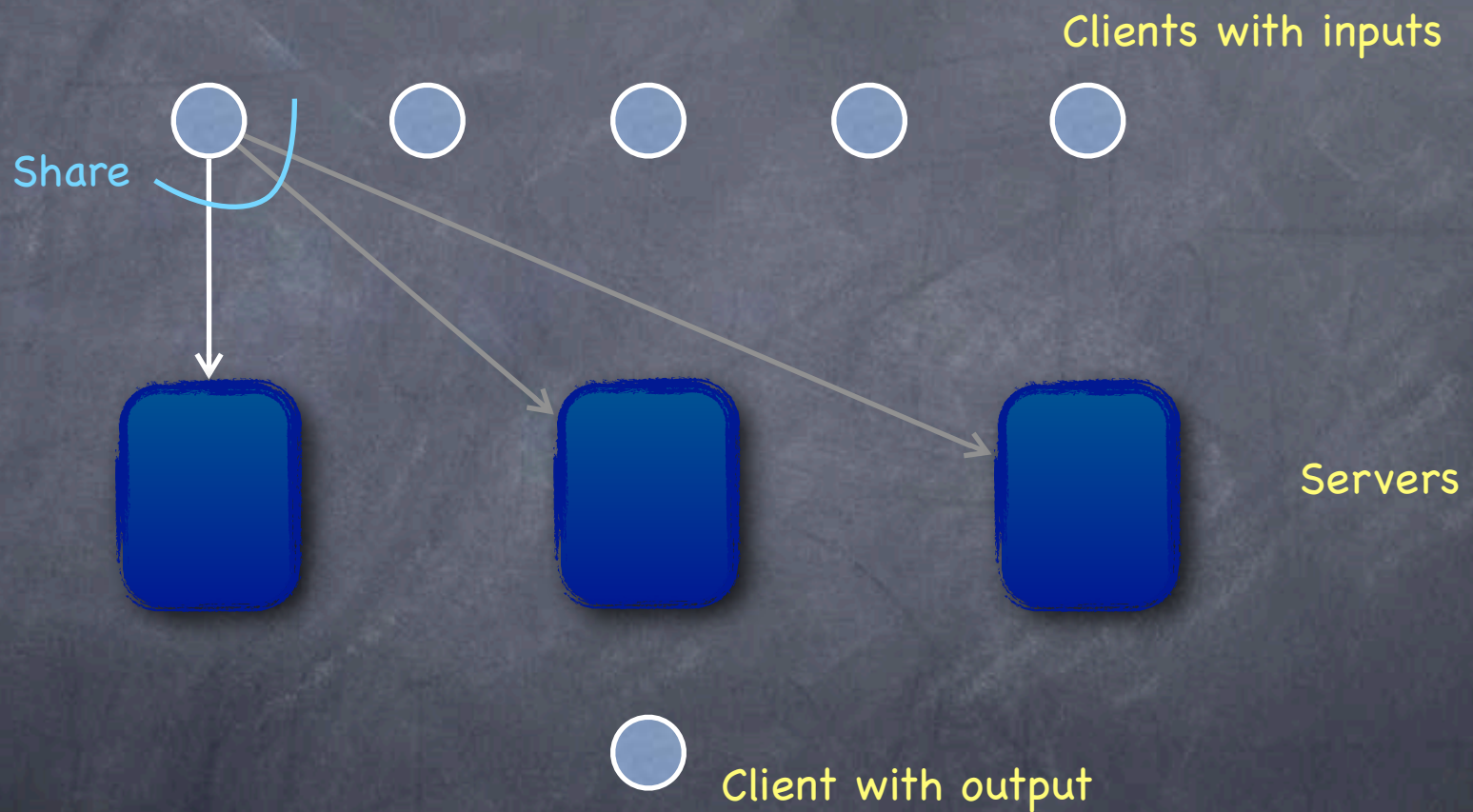
Linear Secret-Sharing

- Gives a "private summation" protocol



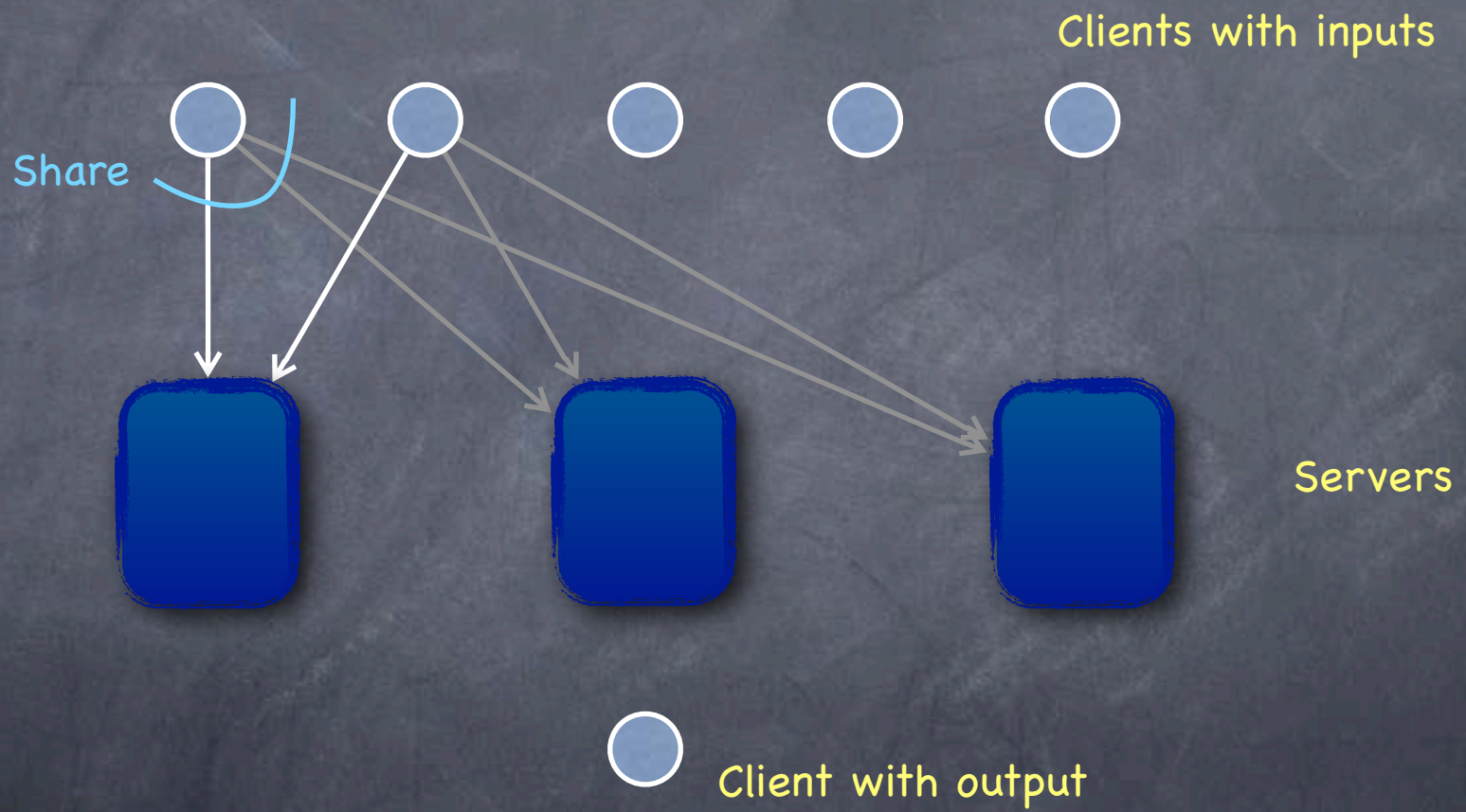
Linear Secret-Sharing

- Gives a "private summation" protocol



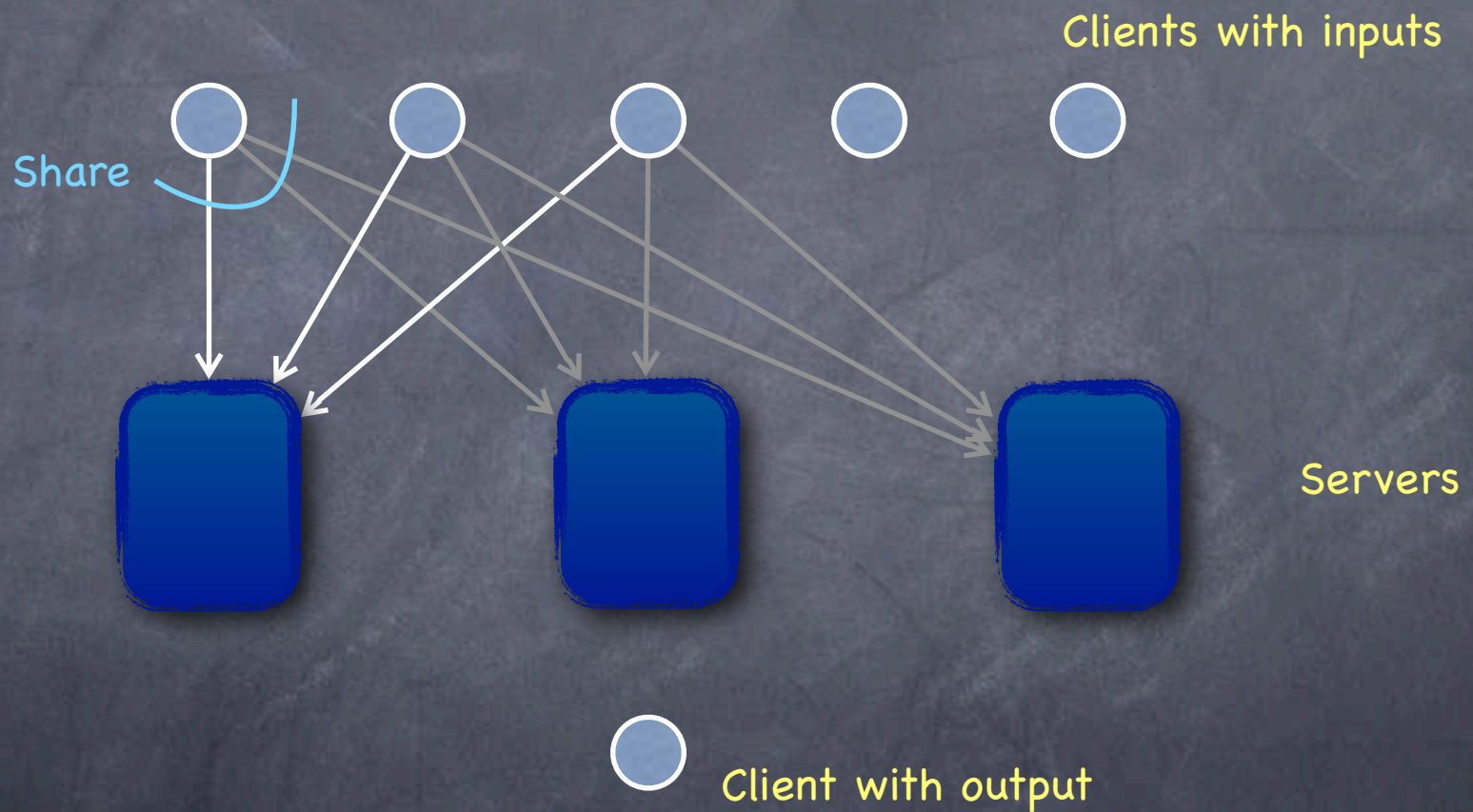
Linear Secret-Sharing

- Gives a "private summation" protocol



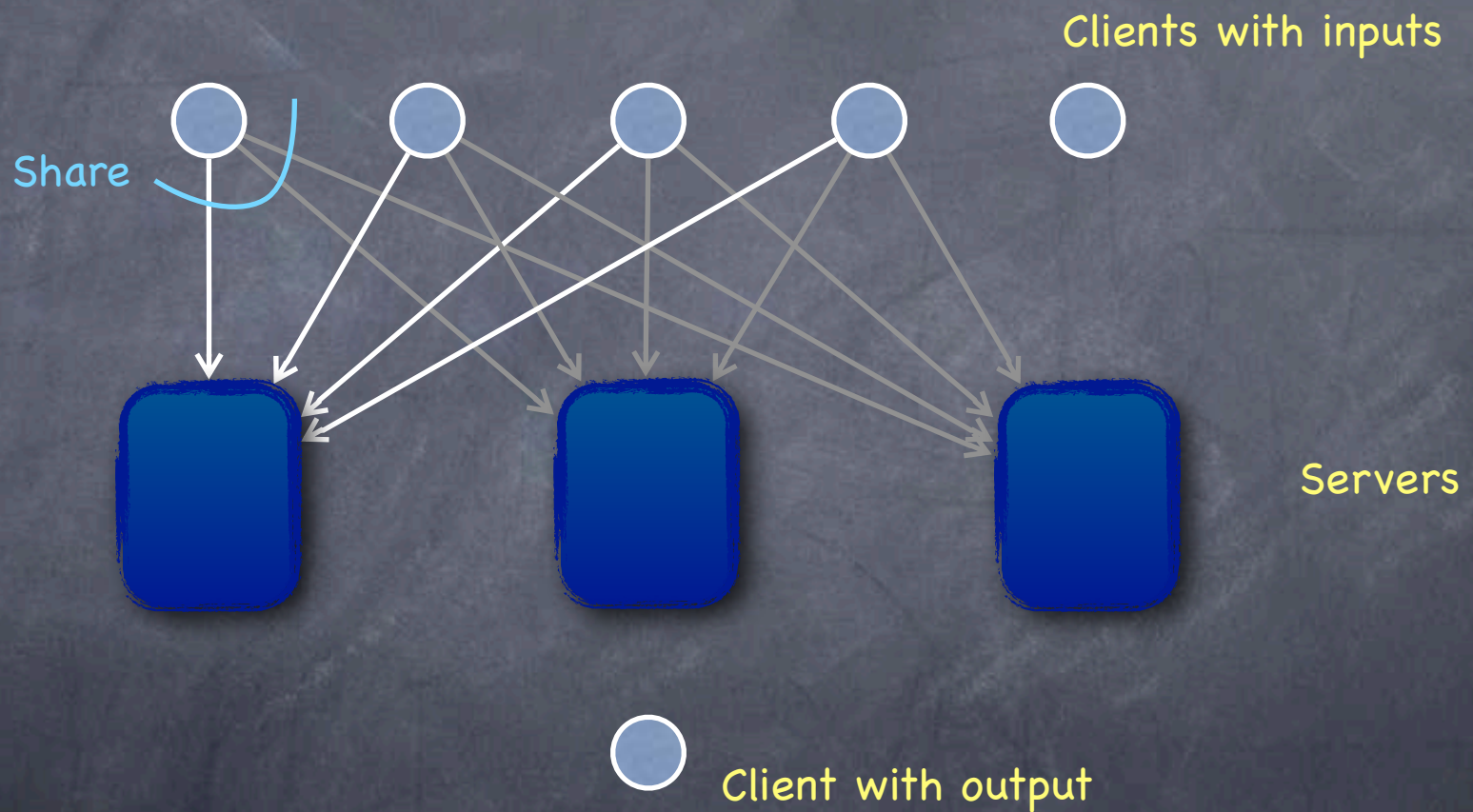
Linear Secret-Sharing

- Gives a "private summation" protocol



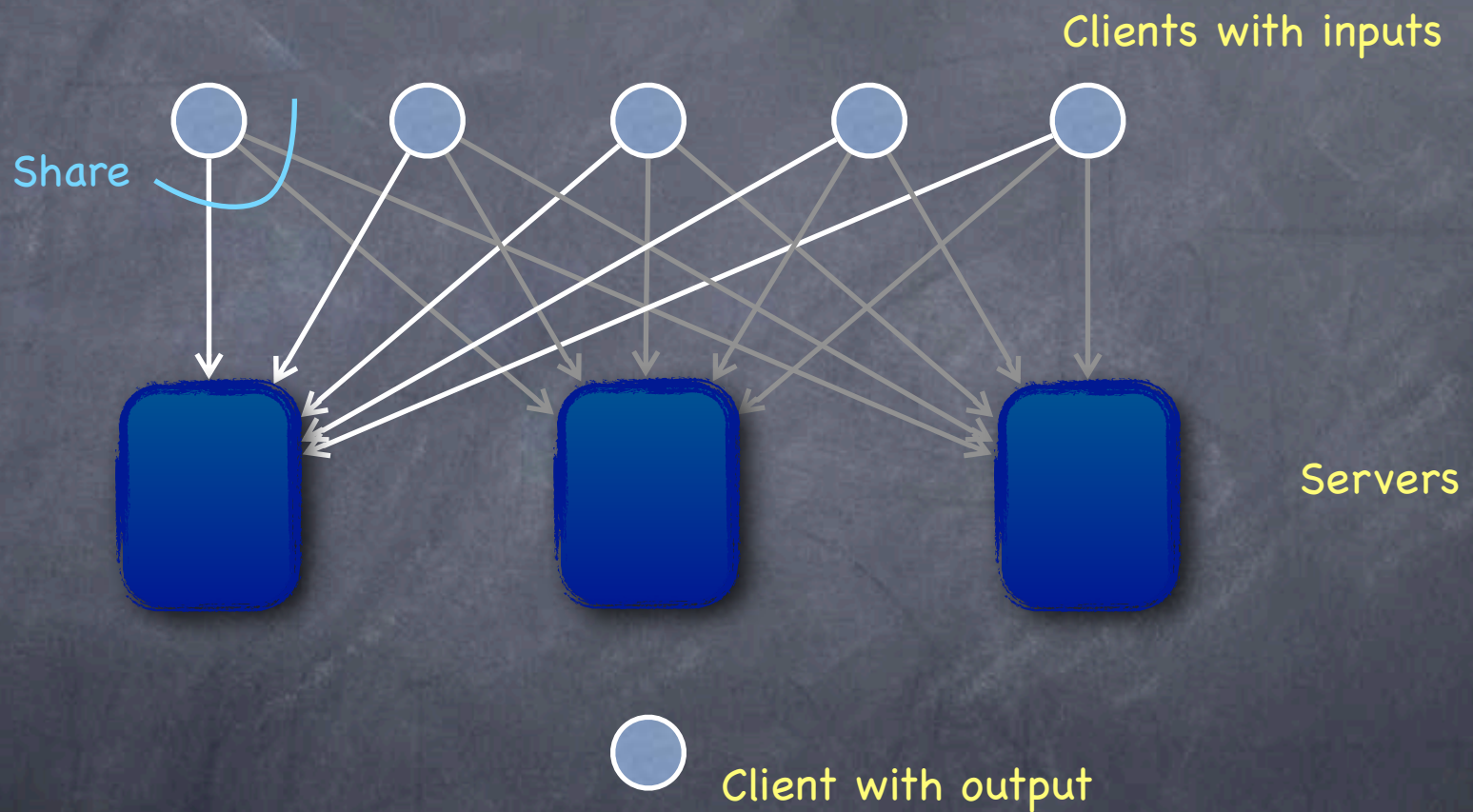
Linear Secret-Sharing

- Gives a "private summation" protocol



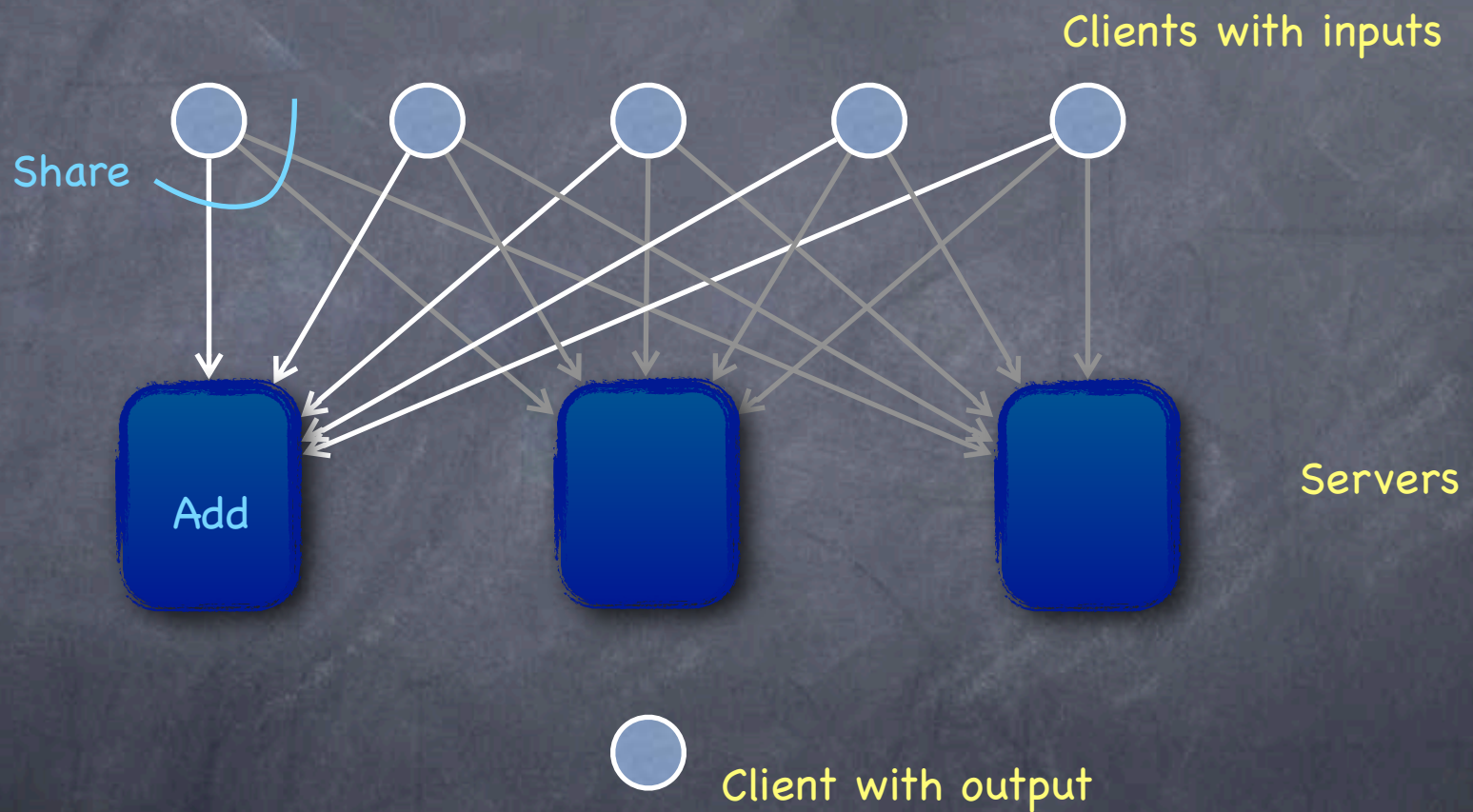
Linear Secret-Sharing

- Gives a "private summation" protocol



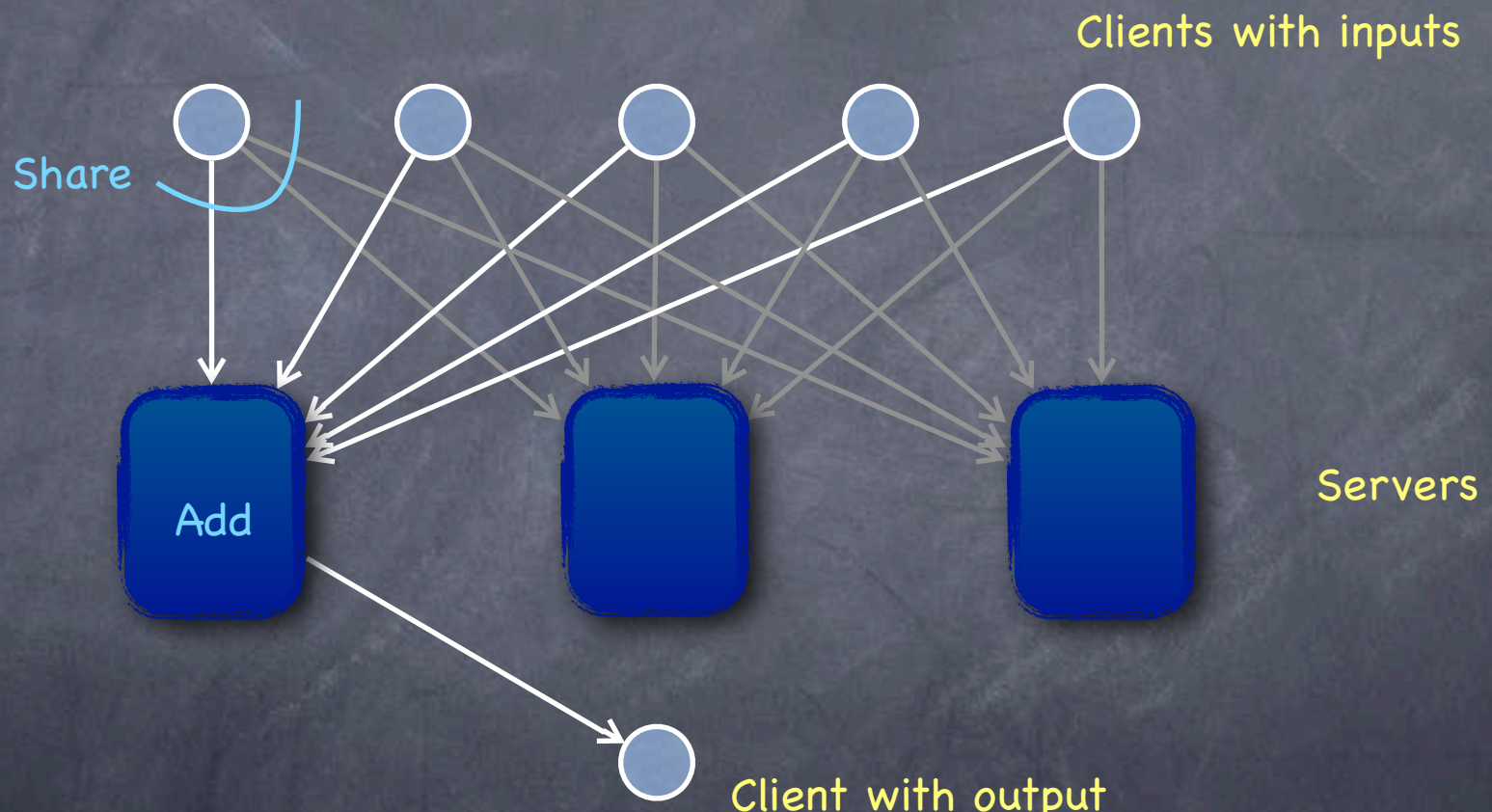
Linear Secret-Sharing

- Gives a "private summation" protocol



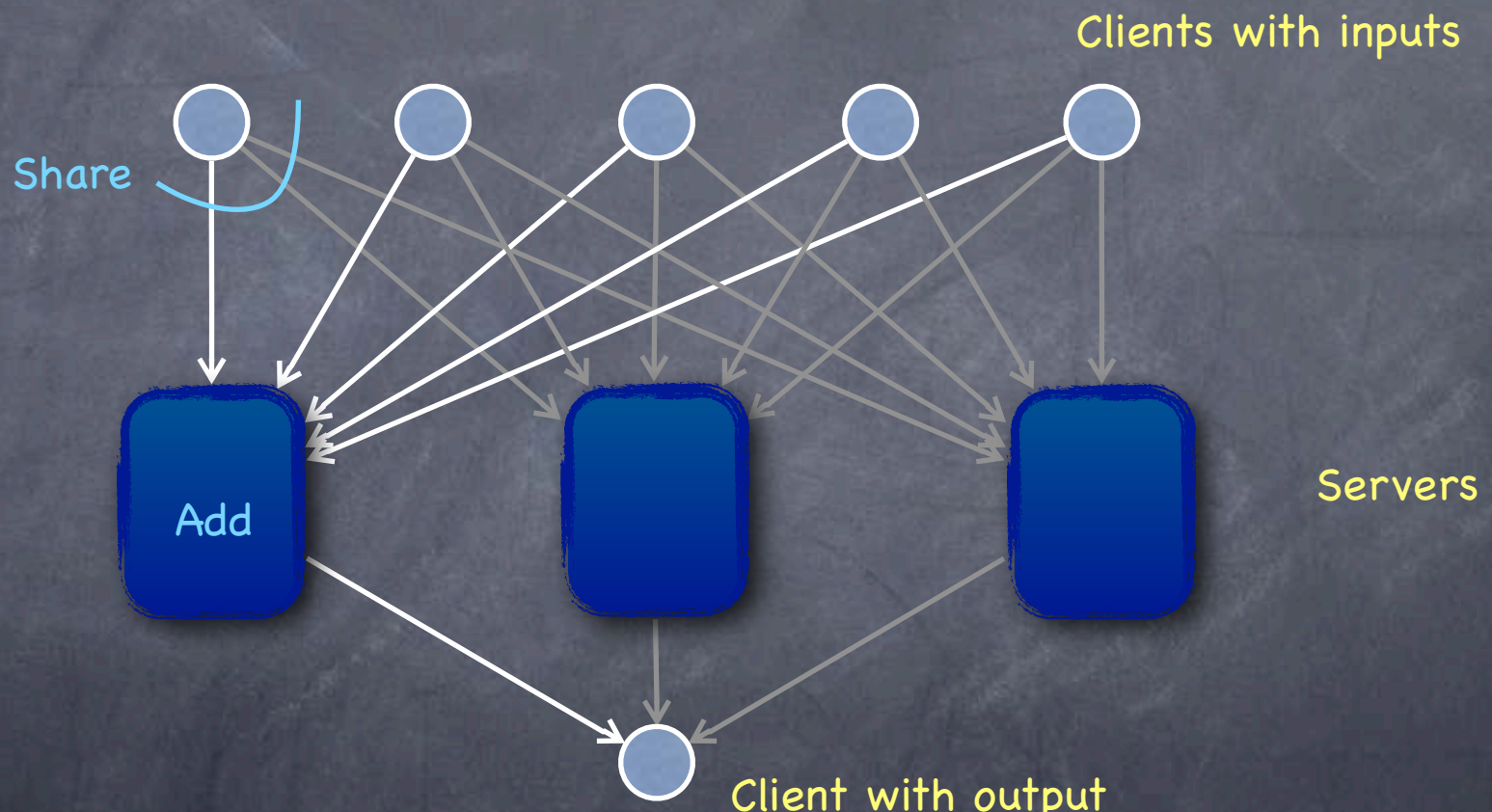
Linear Secret-Sharing

- Gives a "private summation" protocol



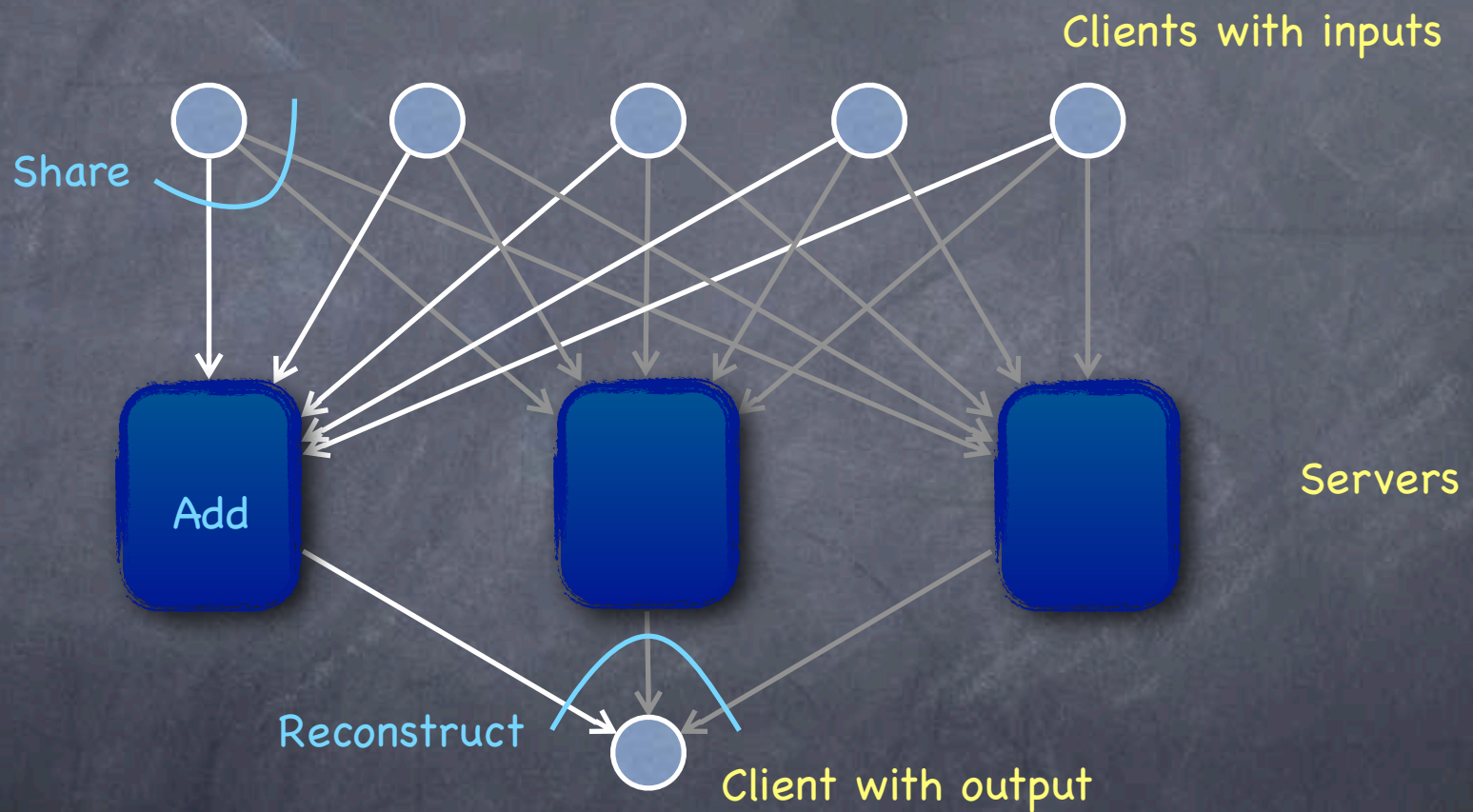
Linear Secret-Sharing

- Gives a "private summation" protocol



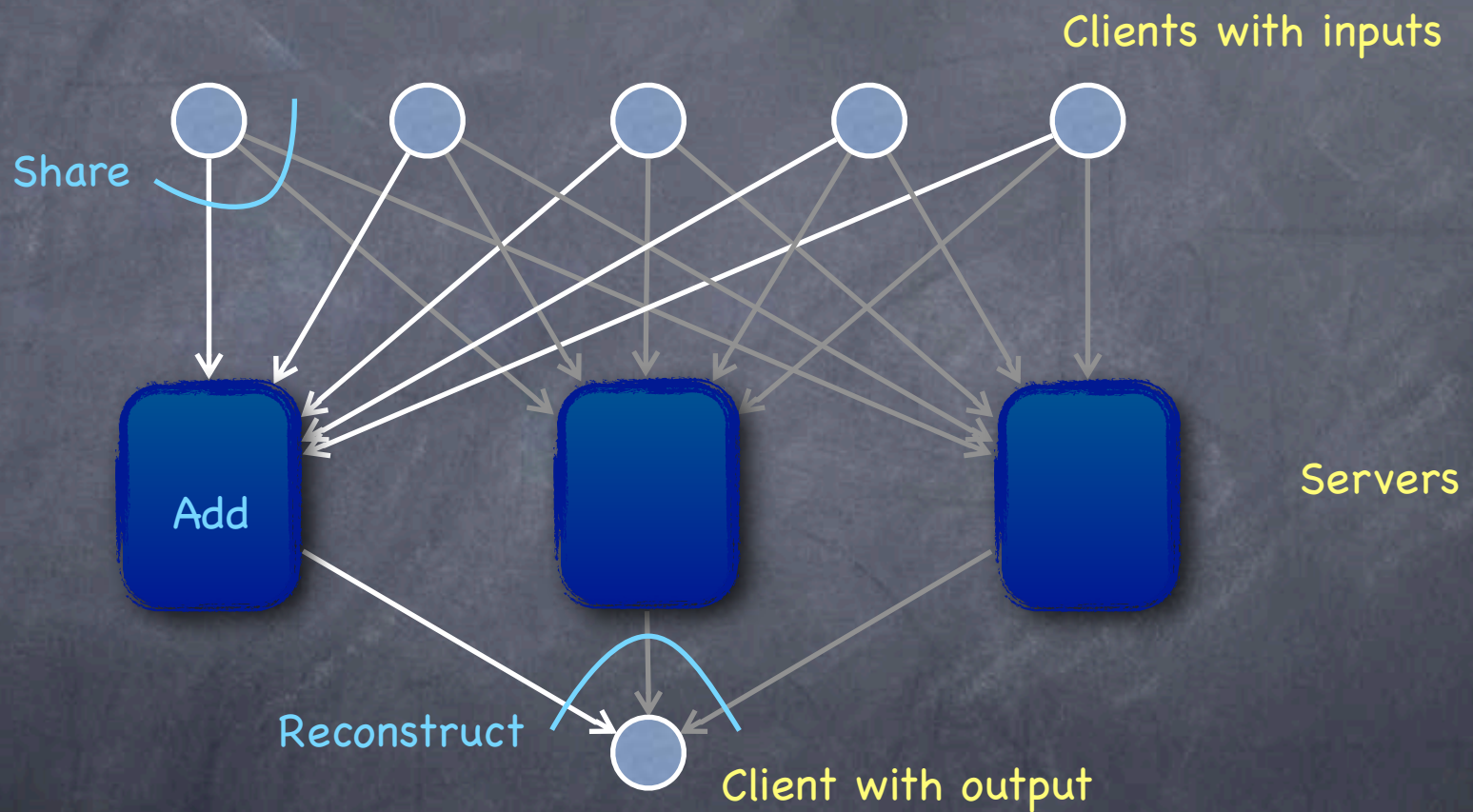
Linear Secret-Sharing

- Gives a "private summation" protocol



Linear Secret-Sharing

- Gives a "private summation" protocol



- Secure against passive corruption (no colluding set of servers/clients learn more than what they must) if at least one server stays out of the collusion

Efficiency

Efficiency

- Main measure: size of the shares (say, total of all shares)

Efficiency

- Main measure: size of the shares (say, total of all shares)
  - Shamir's: each share is as as big as the secret (a single field element)

Efficiency

- Main measure: size of the shares (say, total of all shares)
  - Shamir's: each share is as as big as the secret (a single field element)
  - Naïve scheme for arbitrary monotonic access structure: if a party is in N sets in $\mathcal{B}$, N basic shares

Efficiency

- Main measure: size of the shares (say, total of all shares)
  - Shamir's: each share is as as big as the secret (a single field element)
  - Naïve scheme for arbitrary monotonic access structure: if a party is in N sets in $\mathcal{B}$, N basic shares
    - N can be exponential in n (as $\mathcal{B}$ can have exponentially many sets)

Efficiency

- Main measure: size of the shares (say, total of all shares)
  - Shamir's: each share is as as big as the secret (a single field element)
  - Naïve scheme for arbitrary monotonic access structure: if a party is in N sets in $\mathcal{B}$, N basic shares
    - N can be exponential in n (as $\mathcal{B}$ can have exponentially many sets)
  - Share size must be at least as big as the secret: "last share" in a minimal authorized set should contain all the information about the secret

Efficiency

- Main measure: size of the shares (say, total of all shares)
  - Shamir's: each share is as as big as the secret (a single field element)
  - Naïve scheme for arbitrary monotonic access structure: if a party is in N sets in $\mathcal{B}$, N basic shares
    - N can be exponential in n (as $\mathcal{B}$ can have exponentially many sets)
  - Share size must be at least as big as the secret: "last share" in a minimal authorized set should contain all the information about the secret
    - Ideal: if all shares are only this big (e.g. Shamir's scheme)

Efficiency

- Main measure: size of the shares (say, total of all shares)
  - Shamir's: each share is as as big as the secret (a single field element)
  - Naïve scheme for arbitrary monotonic access structure: if a party is in N sets in $\mathcal{B}$, N basic shares
    - N can be exponential in n (as $\mathcal{B}$ can have exponentially many sets)
  - Share size must be at least as big as the secret: "last share" in a minimal authorized set should contain all the information about the secret
    - Ideal: if all shares are only this big (e.g. Shamir's scheme)
    - Not all access structures have ideal schemes

Efficiency

- Main measure: size of the shares (say, total of all shares)
  - Shamir's: each share is as as big as the secret (a single field element)
  - Naïve scheme for arbitrary monotonic access structure: if a party is in N sets in $\mathcal{B}$, N basic shares
    - N can be exponential in n (as $\mathcal{B}$ can have exponentially many sets)
  - Share size must be at least as big as the secret: "last share" in a minimal authorized set should contain all the information about the secret
    - Ideal: if all shares are only this big (e.g. Shamir's scheme)
    - Not all access structures have ideal schemes
  - Non-linear schemes can be more efficient than linear schemes

Today

Today

- Secrecy: if view is independent of the message

Today

- Secrecy: if view is independent of the message
  - i.e., $\forall$ view, $\forall$ msg_1,msg_2, Pr[view | msg_1] = Pr[view | msg_2]

Today

- Secrecy: if view is independent of the message
  - i.e., $\forall$ view, $\forall$ msg_1,msg_2, Pr[view | msg_1] = Pr[view | msg_2]
  - View does not give any additional information about the message, than what was already known (prior)

Today

- Secrecy: if view is independent of the message
  - i.e., $\forall$ view, $\forall$ msg_1,msg_2, Pr[view | msg_1] = Pr[view | msg_2]
  - View does not give any additional information about the message, than what was already known (prior)
  - Allows for unbounded computational power

Today

- Secrecy: if view is independent of the message
  - i.e., $\forall$ view, $\forall$ msg_1,msg_2, Pr[view | msg_1] = Pr[view | msg_2]
  - View does not give any additional information about the message, than what was already known (prior)
  - Allows for unbounded computational power
- Such secrecy not always possible (e.g., no public-key encryption)

Today

- Secrecy: if view is independent of the message
  - i.e., $\forall$ view, $\forall$ msg_1,msg_2, Pr[view | msg_1] = Pr[view | msg_2]
  - View does not give any additional information about the message, than what was already known (prior)
  - Allows for unbounded computational power
- Such secrecy not always possible (e.g., no public-key encryption)
- Next: secrecy against computationally bounded players