Authenticated Dictionaries with Cross-Incremental Proof (Dis)aggregation

Alin Tomescu[1] @alinush407

Yu Xia[2] @SuperAluex

Zachary Newman[2] zjn@mit.edu

[1]VMware Research, [2]MIT CSAIL

October 28th, 2020

Motivation: Stateless Validation and Beyond

We want authenticated dictionaries (ADs) for:

1. Validation state, which takes **hundreds of GBs** in cryptocurrencies, impeding scalability and decentralization.

- Previous ADs are not sufficiently *updatable*, *aggregatable* or efficient.
- Vector Commitments (VCs) suffice to authenticate validation state, but limit smart contract memory.

2. Transparency logs without extra trust assumptions.

- Previous work [TBP+19, Tom20] uses RSA and bilinear accumulators [BdM94, Ngu05], but has high overheads.

3. Their own sake: non-Merkle ADs are interesting and more powerful.

Our contributions

1. A new notion of cross-incremental proof (dis)aggregation for authenticated data structures,
2. An updatable authenticated dictionary (UAD) construction that supports this notion (and can be used for stateless validation),
3. An append-only authenticated dictionary (AAD) construction that is more efficient and more versatile than previous work (and can be used for transparency logs).

In the process, we also give:

1. New techniques to compute **all** non-membership witnesses across, different (but related) RSA accumulators,
   - Plus, new techniques for aggregating such witnesses.
2. A faster algorithm for witness extraction in Boneh et al's PoKCR protocol [BBF18].

Our ADs and Previous Work

Our ADs and Previous Work

AD scheme	Aggregatable π's?	Binding	Updatability?[1]	Update hint-free?	Non-memb. π's?	Append-only π's?	Prove all fast?
Merkle tree [Mer88]	×	Strong	DI	×	✓	×	✓
SADS [PSTY13]	×	Strong	DI	✓	✓	×	✓
AHTs [PTT16]	×	Weak	×	n/a	✓	×	✓
KVC_1 [BBF18]	One-hop	Strong	DI	×	✓	×	✓
KVC_2 [BBF18]	One-hop	Weak	DI	×	✓	×	✓
AAD [Tom20]	×	Strong	×	n/a	✓	✓	✓
Aardvark [LGG+20]	One-hop	Weak	DI	×	✓	×	×
KVaC [AR20]	One-hop	Weak	DI	✓	×	×	×
Our **UAD**	Cross-incr.	Weak	ADIX	×	✓[2]	✓	✓
Our **AAD**	One-hop	Strong	a[3]DI	×	✓	✓	✓

[1]Of individual proofs (*I*), of aggregated proofs (*A*), of cross-aggregated proofs (*X*) and of digests (*D*).
[2]Our UAD supports non-membership proofs, but they can only be "one-hop" aggregated.
[3]Can only update when existing keys change.

Background

Catalano-Fiore (CF) Vector Commitments [CF13, LM19, CFG⁺20]

Let $\mathbf{v} = [v_1, \ldots, v_n]$. Its digest $\mathsf{dig}(\mathbf{v})$ is:

$$S = g^{\prod_{j\in[n]} e_j}, \text{ where } e_j = H(j) \tag{1}$$

$$\Lambda = \prod_{j\in[n]} (S^{1/e_j})^{v_j} \tag{2}$$

Update $\mathsf{dig}(\mathbf{v})$ after change δ_j at j (given update key S^{1/e_j}):

$$\Lambda' = \Lambda \cdot \left(S^{1/e_j}\right)^{\delta_j} \tag{3}$$

Observations:

- $v_i \in \{0,1\}^\ell$ and $e_i \in \mathsf{Primes}_{\ell+1}$
- Can compute all $(S^{1/e_1}, S^{1/e_2}, \ldots, S^{1/e_n}) \leftarrow \mathit{RootFactor}(g, [e_1, \ldots, e_n])$ [STSY01, BBF18].
- $S = \mathit{RSA.Accumulate}(\{1, 2, \ldots, n\})$ and S^{1/e_j} is an RSA membership witness [BdM94, LLX07].

Subvector Proofs for CF VCs [LM19, CFG+20]

Proof π_I for subvector $\mathbf{v}_I = (v_i)_{i \in I}$, where $I \subseteq [n]$ is $\mathrm{dig}(\mathbf{v} \setminus \mathbf{v}_I)$:

$$S_I = g^{\prod_{j \in [n] \setminus I} e_j} = S^{1/e_I}, \text{ where } e_I = \prod_{i \in I} e_i \tag{4}$$

$$\Lambda_I = \prod_{j \in [n] \setminus I} \left(S_I^{1/e_j} \right)^{v_j} \tag{5}$$

Proof ≈ digest & digest is updatable ⇒ proof is updatable too!

$$\Lambda'_I = \Lambda_I \cdot \left(S_I^{1/e_j} \right)^{\delta_j} \tag{6}$$

Observation: Given S^{1/e_j}, can compute $S_I^{1/e_j} = S^{\frac{1}{e_I e_j}} = ShamirTrick(S_I, S^{1/e_j}, e_I, e_j)$ [Sha81]

Extending the Vector With New Positions [CFG+20]

Add a new position $n+1$ with value v_{n+1} to $\mathsf{dig}(v)$:

$$
\begin{aligned}
S' &= S^{e_{n+1}} && (7)\\
\Lambda' &= S^{v_{n+1}} \Lambda^{e_{n+1}} && (8)\\
&= S^{v_{n+1}} \left(\prod_{j \in [n]} (S^{1/e_j})^{v_j} \right)^{e_{n+1}} = (S'^{1/e_{n+1}})^{v_{n+1}} \prod_{j \in [n]} (S'^{1/e_j})^{v_j} = \prod_{j \in [n+1]} (S'^{1/e_j})^{v_j} && (9)
\end{aligned}
$$

Important: I can do this *sequentially* for m new positions in $O(\ell m)$ time.

(This will be useful in the next slide and in our UAD construction.)

Incrementally Disaggregating Proofs in CF VCs [CFG⁺20]

Disaggregate $\pi_I = (S_I, \Lambda_I)$ for $\mathbf{v}_I$ into $\pi_K = (S_K, \Lambda_K)$ for $\mathbf{v}_K$, where $K \subset I$ and $\Delta = I \setminus K$?

$\mathrm{dig}(\mathbf{v} \setminus \mathbf{v}_I)$ $\mathrm{dig}(\mathbf{v} \setminus \mathbf{v}_K)$

Note that $(\mathbf{v} \setminus \mathbf{v}_I) + \mathbf{v}_\Delta = (\mathbf{v} \setminus \mathbf{v}_I) + (\mathbf{v}_I \setminus \mathbf{v}_K) = \mathbf{v} \setminus \mathbf{v}_K$.

So, for each $i \in \Delta$, sequentially add each e_i to S_I and Λ_I.
Takes $O(\ell|\Delta|)$ $\mathbb{G}_?$ ops, as shown in previous slide.

Implication: Can compute all proofs in $O(\ell n \log n)$ $\mathbb{G}_?$ ops via disaggregation!

(*Campanelli et al. [CFG⁺20] claim* $O(\ell n \log^2 n)$, *but this way appears faster.*)

Incrementally Aggregating Proofs in CF VCs [CFG+20]

Aggregate π_I and π_J into $\pi_{I\cup J}$? (Assume $I \cap J = \varnothing$ or disaggregate.)

$\mathrm{dig}(\mathbf{v} \setminus \mathbf{v}_I)$ & $\mathrm{dig}(\mathbf{v} \setminus \mathbf{v}_J)$ $\mathrm{dig}((\mathbf{v} \setminus \mathbf{v}_I) \setminus \mathbf{v}_J)$

$$S_{I\cup J} = S^{\frac{1}{e_I e_J}} = ShamirTrick(S_I, S_J, e_I, e_J) = ShamirTrick(S^{1/e_I}, S^{1/e_J}, e_I, e_J)$$

$\Lambda_{I\cup J} = \prod_{k\in[n]\setminus(I\cup J)} (S_{I\cup J}^{1/e_k})^{v_k} = \left(\prod_{k\in[n]\setminus(I\cup J)} (S^{1/e_k})^{v_k}\right)^{\frac{1}{e_I e_J}}$ is a bit more complicated. (Two $RootFactor$'s and a $ShamirTrick$ away.)

Observation: $O(\ell |I| \log |I|)$ $\mathbb{G}_?$ ops to aggregate, assuming $|I| > |J|$.

Our Authenticated Dictionary

From CF VC to Authenticated Dictionary

Let D be a dictionary.

Treat D as a “sparse” vector whose indices are its keys. $\mathsf{dig}(D) = (S, \Lambda)$:

$$S = g^{\prod_{k \in D} e_k}, \text{ where } e_k = H(k) \tag{10}$$

$$\Lambda = \prod_{k \in D} (S^{1/e_k})^{v_k} \tag{11}$$

Similar to CF VCs:

- Digest remains updatable (except must handle removing keys).
- Proof π_K for many keys K remains $\mathsf{dig}(D \setminus D(K))$.
- Proof remains updatable (except must handle removing keys).
- Proofs remain incrementally (dis)aggregatable

Concurrent idea with [AR20], which elegantly removes update keys.

We go in a different direction: (1) cross-incremental aggregation, (2) non-membership proofs, (3) strong-binding, and (4) append-only proofs.

Dynamic Authenticated Dictionary: Updating Digest

As previously shown, can easily add $(\hat{k}, v_{\hat{k}})$ to $\mathsf{dig}(D) = (S, \Lambda)$ and get $\mathsf{dig}(D') = (S', \Lambda')$.

Remove $(\hat{k}, v_{\hat{k}})$ from D?
Updated digest $\mathsf{dig}\left(D \setminus D(\hat{k})\right)$ = lookup proof $\pi_{\hat{k}}$ w.r.t $\mathsf{dig}(D)$.

Multiple removals $\hat{K}$?
Updated digest $\mathsf{dig}\left(D \setminus D(\hat{K})\right)$ = aggregated lookup proof $\pi_{\hat{K}}$ w.r.t $\mathsf{dig}(D)$.

Updatable Authenticated Dictionary: Updating Proofs

Proof $\pi_K = \mathsf{dig}(D \setminus D(K)) \Rightarrow$ After adding $(\hat{k}, v_{\hat{k}})$, update to π'_K in the same fashion.

Update to π'_K after removing $(\hat{k}, v_{\hat{k}})$ from D?
π'_K must verify w.r.t. $\mathsf{dig}(D')$, where $D' = D \setminus D(\hat{k})$. Note that:

$$\pi'_K = \mathsf{dig}(D' \setminus D'(K)) = \mathsf{dig}\Big((D \setminus D(\hat{k})) \setminus D'(K)\Big) = \mathsf{dig}\Big((D \setminus D(\hat{k})) \setminus D(K)\Big)$$

But this is exactly $\pi_{K \cup \hat{k}}$ w.r.t. D, which we can aggregate from π_K and $\pi_{\hat{k}}$.

Cross-incremental Proof (Dis)aggregation

m different dictionaries with digest $\mathsf{dig}(D_i) = (A_i, c_i)$:

$$A_i = g^{\prod_{k \in D_i} e_k} \text{ and } c_i = \prod_{k \in D_i} (A_i^{1/e_k})^{v_k} \tag{12}$$

Proofs for K_i w.r.t. $\mathsf{dig}(D_i)$ consists of:

$$W_i = A_i^{1/e_{K_i}} \tag{13}$$

$$\Lambda_i = \left(\prod_{k \in D_i - D_i(K_i)} (A_i^{1/e_k})^{v_k} \right)^{1/e_{K_i}} = \left(\frac{\prod_{k \in D_i} (A_i^{1/e_k})^{v_k}}{\prod_{k \in D_i(K_i)} (A_i^{1/e_k})^{v_k}} \right)^{1/e_{K_i}} \tag{14}$$

$$= \left(c_i / \prod_{k \in K_i} (A_i^{1/e_k})^{v_k} \right)^{1/e_{K_i}} = \alpha_i^{1/e_{K_i}} \tag{15}$$

Can aggregate co-prime roots as $W = \prod_{i \in [m]} W_i$ and $\Lambda = \prod_{i \in [m]} \Lambda_i$ via PoKCR [BBF18], but...

Cross-incremental Proof (Dis)aggregation via PoKCR [BBF18]

...but PoKCR requires $\gcd(e_{K_i}, e_{K_j}) = 1, \forall i \neq j$. Not true if $k \in K_i \cap K_j$, because $e_k | e_{K_i}$ and $e_k | e_{K_j}$.

Fix: Require different $H_i(\cdot)$ for each D_i, so $e_{K_i} = \prod_{k \in K_i} H_i(k)$.
This way, $H_i(k) \neq H_j(k)$ and $\gcd(e_{K_i}, e_{K_j}) = 1, \forall i \neq j$.

Limitation: Each D_i must use different public parameters. Not ideal; future work?

Verifying cross-aggregated proofs

Let $e^* = \prod_{i\in[m]} e_{K_i}$. Verify each W_i aggregated within W (via PoKCR [BBF18]):

$$W^{e^*} \stackrel{?}{=} \prod_{i\in[m]} A_i^{e^*/e_{K_i}} = \mathit{MultiRootExp}((A_i)_{i\in[m]}, (e_{K_i})_{i\in[m]}) \text{ from [BBF18]}$$

If the above holds, then can **extract** all W_i's from W such that $W_i^{e_{K_i}} = A_i$.

- We give *faster* algorithm for this that saves a factor of $O(m/\log m)$ work!

Using the extracted W_i's and a few *RootFactor*'s, we reconstruct the α_i's:

$$\alpha_i = c_i / \prod_{k\in K_i} (A_i^{1/e_k})^{v_k}$$

Then, verify the Λ_i's aggregated within Λ:

$$\Lambda^{e^*} \stackrel{?}{=} \prod_{i\in[m]} \alpha_i^{e^*/e_{K_i}} = \mathit{MultiRootExp}((\alpha_i)_{i\in[m]}, (e_{K_i})_{i\in[m]})$$

If $b = \max_i |K_i|$, verification takes $O(\ell b m(\log^2 m + \log b))$ $\mathbb{G}_?$ ops.

Disaggregating and Updating cross-aggregated proofs

The W_i's and Λ_i's can be extracted from $(W, \Lambda) \Rightarrow$ Can recover original proofs!

Consequences:

- Can disaggregate cross-aggregated proofs
- Can update cross-aggregated proofs

...and that concludes our UAD presentation!

Other goodies for applications beyond stateless validation?

- **Strong binding**: RSA non-membership witness for k w.r.t. S_k from $\pi_k = (S_k, \Lambda_k)$
  - Downgrade to one-hop aggregation (via PoKE [BBF18])
- **Non-membership proofs**: RSA non-membership witness w.r.t. S from $\mathsf{dig}(D) = (S, \Lambda)$
- **Append-only proofs** for transparency logs: observe $S' = S^u$ and $\Lambda' = \Lambda^u S^z$ for $u, z \in \mathbb{Z}$

Conclusion

Catalano-Fiore VCs [CF13] and their extensions [LM19, CFG+20] keep on giving!

- Cross-incremental aggregation (for dictionaries w/ different params)
- ADs with strong-key binding for applications beyond stateless validation
- Append-only proofs
- Non-membership proofs

Be sure to also read [AR20] for how to remove update keys!

What else can we do with CF VCs?

Appendix

Verifying cross-aggregated proofs

Let $e^* = \prod_{i\in[m]} e_{K_i}$ and $b = \max_i |K_i|$. To verify the aggregated W via PoKCR [BBF18]:

$$W^{e^*} \stackrel{?}{=} \prod_{i\in[m]} A_i^{e^*/e_{K_i}} = \mathit{MultiRootExp}((A_i)_{i\in[m]}, (e_{K_i})_{i\in[m]})$$

($\mathit{MultiRootExp}$ takes in $O(\ell bm \log m)$ $\mathbb{G}_?$ ops)

Extract all W_i's from W such that $W_i^{e_{K_i}} = A_i$. (We give faster $O(\ell bm \log^2 m)$ algorithm!)

$\forall i \in [m]$, compute $(A_i^{1/e_k})_{k\in K_i} = \mathit{RootFactor}(W_i, (e_k)_{k\in K_i})$

(Each $\mathit{RootFactor}$ takes $O(\ell b \log b)$ $\mathbb{G}_?$ ops $\Rightarrow$ all take $O(\ell bm \log b)$)

$\forall i \in [m]$, compute $\alpha_i = c_i / \prod_{k\in K_i} (A_i^{1/e_k})^{v_k}$.

$$\Lambda^{e^*} \stackrel{?}{=} \prod_{i\in[m]} \alpha_i^{e^*/e_{K_i}} = \mathit{MultiRootExp}((\alpha_i)_{i\in[m]}, (e_{K_i})_{i\in[m]})$$

Overall, can verify in $O(\ell bm(\log^2 m + \log b))$ $\mathbb{G}_?$ ops.

Incrementally Aggregating Proofs in CF VCs [CFG+20]

Aggregate π_I and π_J into $\pi_{I\cup J}$? (Assume $I \cap J = \varnothing$ or disaggregate.)

$\text{dig}(\mathbf{v} \setminus \mathbf{v}_I)$ & $\text{dig}(\mathbf{v} \setminus \mathbf{v}_J)$ $\qquad$ $\text{dig}(\mathbf{v} \setminus \mathbf{v}_I \setminus \mathbf{v}_J)$

$$S_{I\cup J} = S^{\frac{1}{e_I e_J}} = ShamirTrick(S_I, S_J, e_I, e_J) = ShamirTrick(S^{1/e_I}, S^{1/e_J}, e_I, e_J)$$

$\Lambda_{I\cup J} = \prod_{k\in[n]\setminus(I\cup J)} (S_{I\cup J}^{1/e_k})^{v_k} = \left(\prod_{k\in[n]\setminus(I\cup J)} (S^{1/e_k})^{v_k}\right)^{\frac{1}{e_I e_J}}$ is a bit more complicated.

- Recall $\Lambda_I = \prod_{k\in[n]\setminus I} (S_I^{1/e_k})^{v_k}$.
- Tweak as $\Lambda_I^* = \prod_{k\in[n]\setminus(I\cup J)} (S_I^{1/e_k})^{v_k} = (\prod_{k\in[n]\setminus(I\cup J)} (S^{1/e_k})^{v_k})^{\frac{1}{e_I}}$.
  - How? Divide out all $(S_I^{1/e_k})^{v_k}, k \in J$ from Λ_I
  - How? Compute all $S_I^{1/e_k}, k \in J$ via $RootFactor(S_{I\cup J}, (e_j)_{j\in J})$
- Similarly, $\Lambda_J^* = \prod_{k\in[n]\setminus(I\cup J)} (S_J^{1/e_k})^{v_k} = (\prod_{k\in[n]\setminus(I\cup J)} (S^{1/e_k})^{v_k})^{\frac{1}{e_J}}$.
- Finally, note $\Lambda_{I\cup J} = ShamirTrick(\Lambda_I^*, \Lambda_J^*, e_I, e_J)$

Observation: $O(\ell |I| \log |I|)$ $\mathbb{G}_?$ ops to aggregate, assuming $|I| > |J|$.

Less-efficient Incremental Disaggregation of Proofs in CF VCs [CFG+20]

Disaggregate $\pi_I = (S_I, \Lambda_I)$ for $\mathbf{v}_I$ into $\pi_K = (S_K, \Lambda_K)$ for $\mathbf{v}_K$, where $K \subset I$ and $\Delta = I \setminus K$?

dig$(\mathbf{v} \setminus \mathbf{v}_I)$ dig$(\mathbf{v} \setminus \mathbf{v}_K)$

Note that $(\mathbf{v} \setminus \mathbf{v}_I) + \mathbf{v}_\Delta = (\mathbf{v} \setminus \mathbf{v}_I) + (\mathbf{v}_I \setminus \mathbf{v}_K) = \mathbf{v} \setminus \mathbf{v}_K$.

$$S_K = S_I^{\prod_{j\in\Delta} e_j} = S_I^{e_\Delta} \tag{16}$$

$$\Lambda_K = \prod_{j\in\Delta}\left(S_K^{1/e_j}\right)^{v_j} \Lambda_I^{e_\Delta} \quad \text{(How to get all } S_K^{1/e_j}\text{?)} \tag{17}$$

$$= \prod_{j\in\Delta}\left(S_K^{1/e_j}\right)^{v_j}\left(\prod_{j\in[n]-I}\left(S_I^{1/e_j}\right)^{v_j}\right)^{e_\Delta} \tag{18}$$

$$= \prod_{j\in I-K}\left(S_K^{1/e_j}\right)^{v_j}\prod_{j\in[n]-I}\left(S_K^{1/e_j}\right)^{v_j} = \prod_{j\in[n]-K}\left(S_K^{1/e_j}\right)^{v_j} \tag{19}$$

Observations: Can compute (1) *all* $S_K^{1/e_j}, j \in \Delta$ via $RootFactor(S_I, (e_j)_{j\in\Delta})$ in $O(\ell|I|\log|I|)$ $\mathbb{G}_?$ ops and (2) all proofs in $O(\ell n \log^2 n)$ $\mathbb{G}_?$ ops via disaggregation!

References i

Shashank Agrawal and Srinivasan Raghuraman.
KVaC: Key-Value Commitments for Blockchains and Beyond.
Cryptology ePrint Archive, Report 2020/1161, 2020.
https://eprint.iacr.org/2020/1161.

Dan Boneh, Benedikt Bünz, and Ben Fisch.
Batching Techniques for Accumulators with Applications to IOPs and Stateless Blockchains.
Cryptology ePrint Archive, Report 2018/1188, 2018.
https://eprint.iacr.org/2018/1188.

Josh Benaloh and Michael de Mare.
One-Way Accumulators: A Decentralized Alternative to Digital Signatures.
In Tor Helleseth, editor, *EUROCRYPT '93*, pages 274–285, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

- Dario Catalano and Dario Fiore.
  Vector Commitments and Their Applications.
  In *PKC'13*, 2013.
- Matteo Campanelli, Dario Fiore, Nicola Greco, Dimitris Kolonelos, and Luca Nizzardo.
  Vector Commitment Techniques and Applications to Verifiable Decentralized Storage, 2020.
  https://eprint.iacr.org/2020/149.
- Derek Leung, Yossi Gilad, Sergey Gorbunov, Leonid Reyzin, and Nickolai Zeldovich.
  Aardvark: A Concurrent Authenticated Dictionary with Short Proofs.
  Cryptology ePrint Archive, Report 2020/975, 2020.
  https://eprint.iacr.org/2020/975.

Jiangtao Li, Ninghui Li, and Rui Xue.
Universal Accumulators with Efficient Nonmembership Proofs.
In Jonathan Katz and Moti Yung, editors, *Applied Cryptography and Network Security*, pages 253–269, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

Russell W. F. Lai and Giulio Malavolta.
Subvector Commitments with Application to Succinct Arguments.
In *CRYPTO'19*, 2019.

Ralph C. Merkle.
A Digital Signature Based on a Conventional Encryption Function.
In Carl Pomerance, editor, *CRYPTO '87*, pages 369–378, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

Lan Nguyen.
Accumulators from Bilinear Pairings and Applications.
In Alfred Menezes, editor, *CT-RSA '05*, pages 275–292, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

Charalampos Papamanthou, Elaine Shi, Roberto Tamassia, and Ke Yi.
Streaming Authenticated Data Structures.
In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 353–370, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

Charalampos Papamanthou, Roberto Tamassia, and Nikos Triandopoulos.
Authenticated Hash Tables Based on Cryptographic Accumulators.
Algorithmica, 74(2):664–712, 2016.

References v

Adi Shamir.
On the generation of cryptographically strong pseudo-random sequences.
In Shimon Even and Oded Kariv, editors, *Automata, Languages and Programming*, pages 544–550, Berlin, Heidelberg, 1981. Springer Berlin Heidelberg.

Tomas Sander, Amnon Ta-Shma, and Moti Yung.
Blind, Auditable Membership Proofs.
In Yair Frankel, editor, *Financial Cryptography*, pages 53–71, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

Alin Tomescu, Vivek Bhupatiraju, Dimitrios Papadopoulos, Charalampos Papamanthou, Nikos Triandopoulos, and Srinivas Devadas.
Transparency Logs via Append-Only Authenticated Dictionaries.
In *ACM CCS'19*, CCS '19, page 1299–1316, New York, NY, USA, 2019. Association for Computing Machinery.

Alin Tomescu.
How to Keep a Secret and Share a Public Key (Using Polynomial Commitments).
PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2020.