The Weird World of (Joint) Signatures

Cecilia Boschini

Facoltà Indipendente di Gandria, Gandria, Switzerland

Feeling disheartened by the amount of signatures out there? Do ring signatures give you the creeps? Fear no more: today we are finally putting order in our signatures drawer ☺

Disclaimers. This paper is a dissemination work. As such, it includes a high-level introduction to signatures (Section 1), and a limited amount of cryptographic jargon, with references to papers containing formal definitions for interested readers. For the sake of our sanity, we will not classify *all* the possible signatures and hybrids out there. Instead, we focus on the ones that are currently in the spotlight, i.e., schemes that allow a group of signers to produce a (somewhat) joint signature: multisignatures (with honorable mention of aggregate signatures), threshold signatures, ring signatures and group signatures. Accordingly, this paper does *not* contain: a survey of the different definitions of unforgeability, and MACs. While there might be a followup post about unforgeability definitions[1], a post about MACs is out of the question[2].

TL;DR. Joint signatures come in all shapes and sizes! Sometimes differences are *very* subtle, so we have listed all the schemes mentioned with a short, high-level definition in Table 1.

Acknowledgments. Section 1 has been greatly improved during some ping-pong matches with Xavier Coiteux-Roy and Nicolo Angerer.

Scheme	Description
Multisignatures	Somewhat interactive signing protocols between many parties that yield a joint signature on a message.
Aggregate Signatures	Generalization of multisignatures: they allow to aggregate signatures by different signers on *different* messages.
Ring Signatures	Protocols that allow a signer to anonymously generate a signature on behalf of a ring of signers.
Group Signatures	Ring signatures, but with a group manager that can revoke anonymity.
(t, n)-*Threshold Signatures*	Protocols that require at least t out of the n signers to generate a signature on a message; can be anonymous.

Table 1. Overview of the schemes mentioned in this work.

[1] This is *not* a promise.
[2] This is a promise.

1 Digital Signatures

Digital signatures allow users to sign electronic documents, guaranteeing that nobody can create a signature on behalf of someone else without their[3] explicit consent. But how exactly are they implemented? To understand the cryptographic machinery underlying signatures, let us start from the action they aim to mimic.

1.1 Physical Signing

Signing a physical document guarantees the signer's approval of the document. This is ensured by the hardness of seamlessly physically transferring a signature from a document to another one, and of reproducing someone else's handwriting. In fact, the verification of the authenticity of a signature is similar to assessing the authenticity of a painting. To officially list a newly found painting as a Picasso's, one has to consult a Picasso expert, who verifies that the piece bears all the distinctive signs of Picasso's style by comparing it with other works by him. Similarly, to authenticate a contested (physical) signature, an expert compares it to many other signatures by the same signer to verify its consistency with the signer's handwriting (their "style").

1.2 Digital Signing

When implementing such a functionality for electronic documents, we find ourselves in a pickle: First, we need a digital equivalent of the signer's handwriting (and a way to authenticate it). Second, signatures on electronic documents (think of emails or PDFs) can be easily copy-pasted on other documents: digital signatures should include a way to tie a signature to a file. To address the first issue, a digital-signature scheme equips signers with a signing key and a verification key: The *signing key* is a bit string that the signer keeps secret, and that represents its personal style (handwriting). The *verification key* is a different bit string that is available to the public, and that can be used to verify a signature, corresponding to the handwriting authenticity verification by an expert. A digital signature is then a bit string σ computed from both the secret key *and* the (bit–string representation of) the document to be signed. In this way the signature σ is linked to both: different messages yield different values of σ (as different signing keys would). To verify a signature authenticity requires to get the verification key of the alleged signer, and to check that there is a specific relation between the key, the signature, and the message. This relation is public and depends on the specific scheme. In general, it should be easy to verify, so that anyone that knows the verification key can do it: it turns out that verifying the authenticity of digital signatures is easier than for physical signatures! Exactly as in real life, each verification key is tied to just one signing key: verifying a signature against a different verification key than the signer's never yields that the signature is authentic (in our painting analogy terms: comparing a Picasso's to Raffaello's body of work should not result in classifying the former as part of the latter). This ensures *unforgeability* of a signature: it should be impossible to generate a signature that passes verification without knowing the secret signing key[4].

2 How to Jointly Sign a Document

What happens when there are more than one signers needed to validate a document? In the real world, this is often inconvenient: there should be enough space in the document for all the signatures, which have to be physically collected. Luckily, the digital world offers a variety of protocols that minimize the dimension of the signatures, while requiring different levels of participation from the signers.

2.1 Short Signatures Anyone?

Let us start from the signature length: it would be nice if the signers could come up with just one signature, generated so that it still guarantees that they approve the document. This is what multisignatures[2] do. Multisignatures are a generalization of digital signatures that allow a bunch of users to generate one (somewhat short) joint signature on a message. Verifying such signature requires the verification keys of all the signers: this ensures that everyone actually participated in generating it. Such signatures can be produced either through interactions between the

[3] We use the pronouns they/them as neutral pronouns, both for singular and plural subjects, see `https://www.acm.org/diversity-inclusion/words-matter`.

[4] There are different flavors of unforgeability, depending on which resources the attacker has (e.g., whether the attacker sees signatures generated by the targeted honest signers). For an overview cf. [7].

signers, or by simply aggregating partial signatures produced locally by each signer[5]. Obviously, such schemes guarantee that a group of signers is not able to produce a joint signature unless all members agree.

The Confusing Relative: Aggregate Signatures [5]. Aggregate signatures are often mentioned as a way to achieve short joint signatures. In reality, they focus on a more general problem: can we compress some signatures by different people *on different messages* in just one signature? There are many variant of these signatures, depending on how signatures are aggregated (full, synchronized, or sequential aggregate signatures). Keys are usually generated locally, and generating the partial signatures does not require interactions between signers. However, there might be restrictions on how to compute such partial signatures (e.g., sequential aggregate signatures require the partial signatures to be generated from each other in a specific order). Verification requires all the signed messages and verification keys of the signers to check the signature. As such, aggregate signatures can be seen as a generalization of multisignatures.

2.2 One Signer to Represent Them All

Procedures that require inputs from different parties are usually slow. It would be desirable if a joint signature could be generated by just one of the required signers. Such a signature should not leak the identity of the signer: its goal is just to ensure that the group approved the document.

This is the idea behind group [1,3] and ring signatures [4]: there is a ring (or group) of signers, each of which can generate a signature on behalf of the others, without interacting with them. Both protocols hide the identity of the signer that actually generated a signature (as long as there are more than two signers in the group/ring). Their main difference lies in how they deal with rogue-signers attacks: signer's anonymity could allow a corrupted user to get away with signing unapproved documents on behalf of the group! To detect potential rogue members, anonymity is conditional in group signatures: the scheme includes a (somewhat) trusted party called *group manager* that can "open" a signature to reveal which member of the group generated it. On the other hand, ring signatures remain vulnerable to these attacks: a corrupted user Eve could sign on behalf of herself and Alice, as long as she knows her verification key. Alice has no way to detach herself from it[6].

This difference translates in less flexibility for the group signature, starting from the key generation. While ring signatures allow for each signer to generate their keys by themselves, in a group signature the group manager has to participate in the process, thus generating the keys always requires some level of interaction. The signing process is less flexible too. Ring signatures allow for the ring of signers to be formed only when the signature is generated (as the signer only needs the verification key of the other ring members), and to vary between signatures. On the contrary, a signature generated through a group signature scheme is always on behalf of all the signers that participated in the key generation protocol (i.e., of the entire group). Generating a signature on behalf of a different group requires to run the key generation protocol (or some analogous key-update protocol) again.

2.3 The Optimist Approach: Threshold Signatures

So far we have been pretty pessimist about cosigners, assuming that everyone is trying to frame the one honest signer in the group. However, our daily life is full of nice and trustworthy people! It is a natural question then to ask whether we can relax the security definitions, and assume that there can only be at most $t-1$ misbehaving parties out of the n possible signers. Under this assumption, to prevent rogue-signers attacks it is enough to ensure that a joint signature can only be produced through a collaboration of more than t signers. Such signing protocols are called threshold signatures [9]: given a group of n signers, any subset of more than t of them can generate a signature which is guaranteed to be unforgeable assuming at most $t-1$ rogue signers. This is a compromise, in that it preserves some flexibility in the composition of the group of signers, while still guaranteeing some protection against rogue members.

[5] These are called *non-interactive multisignatures*, which can be confusing to the inexperienced cryptographer: generating the joint signature still requires some kind of interaction, as there has to be a party that collects all the partial signatures and aggregate them together to produce the joint signature. However, such task can be performed by anyone (even the adversary): the only interaction required is when the signers send the partial signatures to the aggregator.

[6] This can be mitigated using a variant of ring signatures called *linkable ring signatures* [6], whose disculpability property allows signers to prove that they did not *produce* a given signature.

Are Threshold Signatures Anonymous? While it being the main goal of group and ring signatures, anonymity does not make sense in the case of aggregate signatures and multisignatures: indeed, in these protocols the signature is not generated by a single signer, but by a subset of them, and verification needs the public keys of all the signers. Threshold signatures are in between these two cases. Traditionally, being anonymous is not required. However, many threshold signature schemes are in fact anonymous (i.e., they hide which subset of the signers actually participated in generating the signature) because of the cryptographic building blocks they rely on (in particular, Shamir secret-sharing and zero-knowledge proofs). This, combined with the need for anonymous threshold signatures in some blockchain constructions, resulted in anonymity being included in the definition of threshold signature in almost all the recent works.

Threshold Signatures Notation. Threshold signatures are often referred to as "(t, n)-threshold signature". However, a careful analysis of the literature combined with this very reliable Twitter poll [8] shows that there is no consensus about what t actually is: Is it the maximum number of misbehaving signers? Or is it the number of signers needed to generate a signature (which is one more than the maximum number of misbehaving signers)? We refer to such notation as *Schroedinger's notation*: the reader cannot know what the notation stands for unless they read the prerequisites.

3 Conclusions

All these terms focus on different approaches to the problem of efficiently generating a joint signature. We have summarized the intuitions behind each term in Table 1. These translate in clear structural differences between the protocols. For the sake of completeness, we have collected these in Table 2 (mostly for the technically inclined reader). Keeping these tables in mind should be enough not to get lost in the signature-related crypto literature ☺

Bonus Content: Signatures vs. MACs. MACs (Message-Authentication Codes) are digital signatures where the same key is used to sign and verify (i.e., $sk = vk$). As such, they have different security definitions and use cases. For formal definitions of MACs see [7, Chapter 6].

References

1. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, Heidelberg, May 2003.
2. Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006.
3. Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, Heidelberg, February 2005.
4. Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 60–79. Springer, Heidelberg, March 2006.
5. Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 416–432. Springer, Heidelberg, May 2003.
6. Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 181–200. Springer, Heidelberg, April 2007.
7. Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.
8. Claudio Orlandi. `https://twitter.com/claudiorlandi/status/1326513421240635400?s=09`, Last accessed on 2021-04-30.
9. Victor Shoup. Practical threshold signatures. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 207–220. Springer, Heidelberg, May 2000.

	Key Generation	Signing Protocol	Unforgeability	Anonymity	Aggregation
Digital signature [7]	local	1 signer	✓	X	X
Multisignature [2]	local/distributed dynamic	any subset of the signers	✓ (at least 1 honest party among signers)	X	✓ public/private only signatures on the same msg
Aggregate Signature [5]	local dynamic	any subset of the signers	✓ (at least 1 honest party among signers)	X	✓ public/private no restrictions on signers/msgs
(t, n)-threshold Signature [9]	centralized/distributed static	t-out-of-n signers required	✓ (at most $t-1$ corrupted signers)	✓ (traditionally, but is not inherent)	X
Ring Signature [4]	local dynamic	1 signer	✓ (if all signers in target ring honest)	(at least 2 honest users in the ring)	X
Group Signature [1,3]	centralized/distributed static or dynamic	1 signer	various notions (traceability, non-frameability)	✓ (revokable)	X

* Homomorphic signature allow to combine signatures by the same signer on different messages, to obtain a valid signature by the same signer on the combination of the messages.

Table 2. Summary of the main differences between the various digital signature schemes considered in this paper. *Local* key generation indicates that each signer generates their key without interacting with other parties (i.e., the plain public-key model [2]). This is in opposition to centralized/distributed key generations, that either required keys to be generated by a trusted party (*centralized*), or through interactions with other parties (*distributed*).

A *dynamic* key generation phase means that signers can join at any point during the protocol execution (even after subsets of signers already started generating signatures). A *static* key generation phase instead implies that signers cannot join after the key generation phase ends.

When not specified, the signing protocol is a confidential protocol run by the signer/s.

Aggregating signatures is *private* if it can only be done by the signers, otherwise is *public*.