Legally Fair Contract Signing Without Keystones

Paper accepted at ACNS 2016 (to appear)

Houda Ferradi, Rémi Géraud, Diana Maimuț, David Naccache, and David Pointcheval

Ecole normale supérieure, Paris, France

houda.ferradi@ens.fr

ÉCOLE NORMALE SUPÉRIEURE

Abstract

In two-party computation, achieving both fairness and guaranteed output delivery is well known to be impossible. In this paper we describe and analyse a new contract signing paradigm concept called *legal fairness*. This paradigm is very close to fairness and is realizable. We give a concrete legal fairness protocol based on Schnorr signatures. The new protocol is provably secure in the random oracle model under the DLP assumption.

Introduction

When mutually distrustful parties wish to compute some joint function of their private inputs, they require a certain number of security properties to hold for that computation:

- *Privacy*: Nothing is learnt from the protocol besides the output;
- *Correctness*: The output is distributed according to the prescribed functionality;
- *Independence*: One party cannot make their inputs depend on the other parties' inputs;
- *Delivery*: An adversary cannot prevent the honest parties from successfully computing the functionality;
- *Fairness*: If one party receives output then so do all.

Any multi-party computation can be securely computed as long as there is a honest majority. If there is no such majority, and in particular in the two-party case, it is impossible to achieve both fairness and guaranteed output delivery.

Objectives

We describe a new contract signing protocol that achieves a new notion of fairness and abuse-freeness. This protocol is based on the well-known Schnorr signature protocol. The new contract signing protocol is provably secure in the random oracle model under the hardness assumption of solving the discrete logarithm problem. This construction can be adapted to other DLP schemes.

Schnorr Signatures

Schnorr digital signatures are an offspring of ElGamal signatures. To generate signature keys select large primes p, q such that $p - 1 \bmod q = 0$, as well as an element $g \in \mathbb{G}$ of order q in some multiplicative group $\mathbb{G}$ of order p, and a hash function $H : \{0,1\}^* \to \{0,1\}^\ell$. The output is a set of public parameters $\mathsf{pp} = (p, q, g, \mathbb{G}, H)$. Select at random $x \xleftarrow{\$} \mathbb{Z}_q^\times$ and compute $y \leftarrow g^x$. The output is the couple $(\mathsf{sk}, \mathsf{pk})$ where $\mathsf{sk} = x$ is kept private, and $\mathsf{pk} = y$ is made public.

To sign a message m select a random $k \xleftarrow{\$} \mathbb{Z}_q^\times$ and compute:

$$r \leftarrow g^k, \qquad e \leftarrow H(m\|r), \qquad s \leftarrow k - ex \bmod q$$

and outputs $\langle r, s\rangle$ as the signature of m.

To verify the signature compute $e \leftarrow H(m, r)$ and check that $g^s y^e = r$.

Schnorr Co-Signatures

Schnorr's signatures can be generalized to two signers. This produces co-signatures, i.e. a signature formed by joining forces between two signers:

Alice		*Bob*
$y_{A,B} \leftarrow y_A y_B$ $k_A \xleftarrow{\$} \mathbb{Z}_q^*$ $r_A \leftarrow g^{k_A}$		$y_{A,B} \leftarrow y_A y_B$ $k_B \xleftarrow{\$} \mathbb{Z}_q^*$ $r_B \leftarrow g^{k_B}$ $\rho \leftarrow H(0\|r_B)$
	$\xleftarrow{\ \rho\ }$	
	$\xrightarrow{\ r_A\ }$	
	$\xleftarrow{\ r_B\ }$	
if $H(0\|r_B) \neq \rho$ **abort** $r \leftarrow r_A r_B$ $e \leftarrow H(1\|m\|r)$ $s_A \leftarrow k_A - e x_A \bmod q$		$r \leftarrow r_A r_B$ $e \leftarrow H(1\|m\|r)$ $s_B \leftarrow k_B - e x_B \bmod q$
	$\xleftarrow{\ s_B\ }$	
	$\xrightarrow{\ s_A\ }$	
$s \leftarrow s_A + s_B \bmod q$		$s \leftarrow s_A + s_B \bmod q$

Note that during the co-signature protocol, A might decide not to respond to B: In that case, A would be the only one to have the complete co-signature. This is a breach of fairness insofar as A can benefit from the co-signature and not B, but the protocol is abuse-free: A cannot use the co-signature as a proof that B, and B alone, committed to m. **This is what we seek to fix with a new notion called "legal fairness"**

Legal Fairness

The main idea builds on the following observation: Every signature exchange protocol is plagued by the possibility that the last step of the protocol is not performed. Indeed, it is in the interest of a malicious party to get the other party's signature without revealing its own. As a result, the best one can hope for is that a trusted third party can eventually restore fairness.

To avoid this destiny, the proposed paradigm, called **Legal Fairness**, does *not* proceed by sending A's signature to B and vice versa. Instead, we construct a *joint signature*, or *co-signature*, of both A and B. By design, there are no signatures to steal — and stopping the protocol early does not give the stopper a decisive advantage.

How Does It Work?

We now address a subtle weakness in the protocol described in the previous section, which is not captured by the fairness property *per se* and that we refer to as the existence of "proofs of involvment". Such proofs are not valid co-signatures, and would not normally be accepted by verifiers, but they nevertheless are valid evidence establishing that one party committed to a message. In a legally fair context, it may happen that such evidence is enough for one party to win a trial against the other — who lacks both the co-signature, and a proof of involvment.

To enforce fairness on the co-signature protocol, we ask that the equivalent of a keystone is transmitted first; so that in case of dispute, the aggrieved party has a legal recourse. First we define the notion of an authorized signatory credential:

definition[Authorized signatory credential] The data field

$$\Gamma_{\text{Alice,Bob}} = \{\text{Alice}, \text{Bob}, k_A, \sigma_{x_A}(g^{k_A}\|\text{Alice}\|\text{Bob})\}$$

is called an *authorized signatory credential* given by Alice to Bob, where σ_{x_A} is some publicly known auxiliary signature algorithm using Alice's private key x_A as a signing key.

Any party who gets $\Gamma_{\text{Alice,Bob}}$ can check its validity, and releasing $\Gamma_{\text{Alice,Bob}}$ is *by convention* functionally equivalent to Alice giving her private key x_A to Bob. A valid signature by Bob on a message m exhibited with a valid $\Gamma_{\text{Alice,Bob}}$ is *legally* defined as encompassing the meaning ($\Rrightarrow$) of Alice's signature on m:

$$\{\Gamma_{\text{Alice,Bob}}, \text{signature by Bob on } m\} \Rrightarrow \text{signature by Alice on } m$$

Second, the co-signature protocol is modified by requesting that Alice provide $t = \sigma_{x_A}(g^{k_A}\|\text{Alice}\|\text{Bob})$ to Bob. Bob stores this in a local non-volatile memory $\mathcal{L}$ along with s_B. For all practical purposes, $\mathcal{L}$ can be simply regarded as Bob's hard disk. Together, t and s_B act as a keystone enabling Bob (or a verifier, *e.g.* a court of law) to reconstruct $\Gamma_{\text{Alice,Bob}}$ if Alice exhibits a (fraudulent) signature binding Bob alone with his co-signing public key.

Therefore, should Alice try to exhibit a signature of Bob alone on a message they both agreed upon (which is known as a fraud), the court would be able to identify Alice as the fraudster. The resulting protocol is:

Alice		*Bob*
$y_{A,B} \leftarrow y_A y_B$ $k_A \xleftarrow{\$} \mathbb{Z}_q^*$ $r_A \leftarrow g^{k_A}$		$y_{A,B} \leftarrow y_A y_B$ $k_B \xleftarrow{\$} \mathbb{Z}_q^*$ $r_B \leftarrow g^{k_B}$ $\rho \leftarrow H(0\|r_B)$
	$\xleftarrow{\ \rho\ }$	
$t \leftarrow \sigma(r_A\|\text{Alice}\|\text{Bob})$	$\xrightarrow{\ r_A, t\ }$	
		if t is incorrect **then abort** store t in $\mathcal{L}$
	$\xleftarrow{\ r_B\ }$	
if $H(0\|r_B) \neq \rho$ **then abort** $r \leftarrow r_A r_B$ $e \leftarrow H(1\|m\|r\|\text{Alice}\|\text{Bob})$ $s_A \leftarrow k_A - e x_A \bmod q$		$r \leftarrow r_A r_B$ $e \leftarrow H(1\|m\|r\|\text{Alice}\|\text{Bob})$ $s_B \leftarrow k_B - e x_B \bmod q$ store s_B in $\mathcal{L}$
	breakpoint 1 $\xleftarrow{\ s_B\ }$	
if s_B is incorrect **then abort**		
	breakpoint 2 $\xrightarrow{\ s_A\ }$	
		if s_A is incorrect **then abort**
	breakpoint 3	
$s \leftarrow s_A + s_B \bmod q$		$s \leftarrow s_A + s_B \bmod q$ **if** $\{m, r, s\}$ is valid **then** erase t, s_B from $\mathcal{L}$

And the corresponding dispute resolution algorithm is: