A Novel Multi-factor ID-based Designated Verifier Signature scheme

Michael Scott

Cryptographic Researcher
MIRACL Labs
mike.scott@miracl.com

Abstract. In a classic digital signature scheme, the global community is capable of verifying a signature. In a designated verifier scheme (DVS), only the designated verifier has this capability. In a classic DVS scheme the signer themselves "designates" the entity that will have the capability of verifying their signature. In a pure identity-based signature scheme a Trusted Authority is introduced, and is responsible for issuing secret signing keys to all participants. In our proposed scheme it is this TA, not the signer, that designates the verifier, and to this end the TA issues the designated verifier with its own secret. Finally we propose a variation that supports non-repudiation, plus a hardware-free multi-factor signature capability.

1 Introduction

Consider a scenario where a Bank wishes to have the capability of verifying customer signatures on important transactions. It is not important or indeed desired that the general public can verify these signatures - it is sufficient that the Bank is convinced of their validity, and that disputes can be legally resolved if necessary.

In our proposed solution a Distributed Trusted Authority (D-TA) infrastructure will be required which confirms identities and issues shares of the secret keys. The Bank itself would naturally control one component of this infrastructure, but one or more external identity provisioners should be involved as well.

First we design a simple system. Then we add a non-repudiation feature (not normally possible with Identity-Based signature). Finally we show how such a signature method can be made "multi-factor", so for example a customer can sign using a software token and a memorised PIN number. We would suggest that this is a better solution than currently deployed alternatives, which typically require the bank to issue an expensive hardware device to each customer.

2 A Basic Scheme

Our basic scheme is based on a Identity-Based signature proposal by Cha and Cheon [6] (and independently by Yi [11]), based on type-1 bilinear pairings on elliptic curves. The most common types of pairing are the so-called type-1 and type-3 pairings [8]. A type-1 pairing $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ maps a pair of points in a prime order elliptic curve group $\mathbb{G}_1$ of order q, to an element of order q in a finite extension field $\mathbb{G}_T$. A type-3 pairing $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ does the same, but takes its inputs from points in two different elliptic curve groups $\mathbb{G}_1$ and $\mathbb{G}_2$, of the same order q.

The standard notion for security in this signature context context is unforgeablity under chosen message attack. Cha and Cheon were able to prove this in the random oracle model assuming the hardness of the Computational Diffie-Hellman (CDH) problem in the group $\mathbb{G}_1$ [6]. See also [5]. For our purposes we need to shift from a type-1 pairing to the more efficient type-3 pairing context, which in turn requires a change in the hardness of the underlying assumption, from CDH to co-CDH* [7]. This later assumption takes into account the involvement of points from two separate elliptic curve groups.

- *Setup.* A Trusted Authority (TA) chooses a type-3 pairing friendly elliptic curve and publishes its parameters. The TA generates a random secret s for use with a particular designated verifier and a fixed public generator $Q \in \mathbb{G}_2$. A hash function $H(.)$ is provided which hashes and maps identity strings to a point in $\mathbb{G}_1$. A second hash function $h(.)$ hashes an input of arbitrary length into a digest of size less than q.
- *Key Extract - verifier.* The secret key issued to the designated verifier by the TA is the point $sQ \in \mathbb{G}_2$.
- *Key Extract - signer.* For a signer Alice with identity ID_a, the associated ID-based public key is $A = H(ID_a)$. The secret key issued by the TA to Alice is the point $sA \in \mathbb{G}_1$.
- *Sign.* To sign a message m, Alice generates a random x, calculates $A = H(ID_a)$, $U = xA$, $V = -(x + h(m, U))sA$, and the signature as the tuple (ID_a, U, V).
- *Verify.* To verify Alice's signature on m calculate $A = H(ID_a)$ and $g = e(V, Q).e(U + h(m, U)A, sQ)$, and if $g = 1$ accept the signature, otherwise reject it.

This simple system has a number of short-comings. Clearly the TA is in a position to create a signature on any message, and thus signatures might be repudiated by exploiting a perceived key-escrow issue. One partial solution to this problem is to distribute the TA functionality, and establish a D-TA. For example a pair of D-TAs can generate their own independent secrets s_1 and s_2, and separately issue s_1A and s_2A to Alice, and s_1Q and s_2Q to the designated verifier. Both clients and verifier simply add these components together to form their full secrets.

Nevertheless a conspiracy of D-TAs can still forge signatures. To provide full non-repudiation, we turn to ideas from Certificateless Cryptography [3].

3 A scheme with non-repudiation

The setup and verifier key extraction is as above. We highlight the changes in red.

- *Key Extract - signer.* Alice generates a random z, and calculates a non-ID-based public key $P_a = z^{-1}Q$, and presents the composite identity $ID_a|P_a$ to the trusted authority, which constructs the ID-based public key as $A = H(ID_a|P_a)$, and returns the secret key $sA \in \mathbb{G}_1$. Alice then modifies her secret key to be zsA, and deletes z and sA.
- *Sign.* To sign a message m, Alice generates a random x, calculates $A = H(ID_a|P_a)$, $U = xA$, $V = -(x + h(m, U))zsA$, and the signature as the tuple (ID_a, P_a, U, V).
- *Verify.* To verify the signature on m, independently calculate $A = H(ID_a|P_a)$. Next calculate $g = e(V, P_a).e(U + h(m, U)A, sQ)$. If $g = 1$ accept the signature, otherwise reject it.

Certificateless cryptography [3] does not increase the number of secrets to be protected compared with pure Identity-Based cryptography, but it does introduce a new individualised public key P_a. It is assumed that the TA (or equivalently at least one component of a D-TA) will not get involved in an attack involving replacement of these public keys. The authors of [3] argue that this is essentially the same level of trust we have in a PKI Certificate Authority (CA) not to issue false certificates. Hence their schemes are "certificateless", as public keys gain no extra security from being embedded into a certificate. For digital signatures to have the non-repudiation property, an assumption of this kind is regarded as acceptable, and does not invalidate the legality of PKI signatures. Therefore here we assume that at least one of our independent D-TAs has the same level of integrity as a CA.

4 A multi-factor variation

Multi-factor methods of signature are often required by organisations like Banks, and normally require that customers be provisioned with an expensive hardware token. Here we suggest a cheaper software-only solution.

Consider the possibility of splitting a private key used for digital signature into two parts, say a 4-digit PIN number and a software token. Both components would need to be recombined in order to form a signature. In a normal digital signature scheme clearly this will not work – an attacker who captures the software token simply has to try every possible PIN number until the private key formed by combining token and PIN matches the user's public verification key.

But in our ID-based designated verifier scheme, the single verification key is in fact now a secret known only to the designated verifier. So a software-only 2-factor signature becomes a possibility.

Consider the scenario where an attacker has access to the token and a valid signature. Is this sufficient to find the PIN via an off-line attack? Consider again our basic scheme. Now choose a PIN number α and break the secret into the token $sA - \alpha A$ and the PIN derived component αA, which must be recombined by simple addition to affect a signature.

Definition 1. (XDH assumption [10], [4]) *The Decisional Diffie-Hellman assumption in the group* $\mathbb{G}_1$*: Given the five group elements* $P, aP, bP, abP, cP \in \mathbb{G}_1$*, distinguish between the tuples* $\{aP, bP, abP\}$ *and* $\{aP, bP, cP\}$*, for random* $a, b, c \in \mathbb{F}_q$

Theorem 1. *An adversary who has captured a token and a valid signature cannot determine the PIN, assuming that the XDH problem is hard.*

Proof. Let $c = h(m, U)$ and $A = H(ID_a)$. Assume the existence of an Oracle which can solve this problem: When presented with the input tuple $\{A, xA, c, (x + c)sA, (s - \alpha)A\}$ derived from a signature on m and a captured token, it responds with the correct value of α.

We will show that such a capability can be used to solve the decisional Diffie-Hellman problem in the group $\mathbb{G}_1$. That this problem is hard is the XDH assumption.

Recall that the decisional Diffie-Hellman problem is to distinguish with probability greater than chance the input distributions $\{xA, yA, zA\}$ and $\{xA, yA, xyA\}$ where x, y, z are random elements in $\mathbb{F}_q$.

Given access to our Oracle this decisional problem can be solved by generating a random $c < q$ and $\alpha < 10^4$, and submitting the tuple $\{A, xA, c, zA + cyA, yA - \alpha A\}$ to the Oracle.

If $z = xy$, then this tuple is $\{A, xA, c, (x + c)yA, (y - \alpha)A\}$, and our Oracle will return the same α, and we can report that the input represents a valid Diffie-Hellman triple.

On the other hand if $z \neq xy$, then the Oracle will either fail (as it cannot find a PIN within the expected 4-digit range) and return $\perp$, or return the wrong value for α, $\alpha' = \alpha + (z - xy)/(x + c) \bmod q$, in which cases we report that the input is not a valid Diffie-Hellman triple.

Since the XDH problem is assumed to be hard, we must assume that no such Oracle exists, and therefore the PIN is safe from an off-line dictionary attack.

The same idea can be easily extended to our scheme that supports non-repudiation, with the client secret split into the token $zsA - \alpha A$ and the PIN derived component αA. However we observe that this idea will not work with other Identity-Based signature schemes. For example if applied to the scheme due to Hess [9] considered in [3] the PIN can be discovered via an off-line dictionary attack.

The extension of the idea from 2-factor to multi-factor is straightforward.

5 Discussion

A conspiracy of D-TAs can re-create sQ, and therefore can verify signatures. As a consequence, such a conspiracy that gains access to client tokens can calculate the associated PINs. We do not believe that this constitutes a realistic attack, requiring as it does the subversion of all of the D-TAs and breaches of individual client security.

While a conspiracy of D-TAs can forge individual signatures, the security risk is equivalent to that posed by a corrupt Certificate Authority in a PKI signature context. Recall that in this situation the existence of a pair of digital signatures on the same message associated with the same identity, but with different public keys, suffices to expose such a conspiracy [3].

Revocation can be supported either as described in [3], or using the concept of “time permits”, which are publicly issued by the D-TAs on, for example, a daily basis. See below.

6 Application

It has been assumed that as part of the solution to the challenge of secure multi-factor authentication, some kind of hardware vault like a Trusted Platform Module (TPM) is required on the client side. Typical solutions such as that proposed by the FIDO alliance [1] require the protection of a PKI-like signing key on the client device, which is used to sign a random challenge issued by the server every time the client wishes to authenticate. Ideally for an optimal user experience this signing key should be activated from the hardware vault by the entry of a low entropy secret like a PIN number, or a biometric which matches a template stored inside the vault. We should point out that this process is two-step rather than two-factor. An attacker who can gain access to the TPM, either by successful hacking, or by way of a manufacturer’s back-door, can gain immediate access to the secret, with no need of a PIN or a biometric. As a bonus they might capture the biometric template as well. It would appear not to be possible to conveniently implement such functionality entirely in software on the client side. As is well known the cryptographic protection of a PKI private key in software requires the use of a very user-unfriendly high-entropy pass-phrase, a process made famous by PGP [2].

Our proposed solution is genuinely two-factor/multi-factor. Furthermore a software only implementation on the client side is now perfectly feasible, which reduces the cost dramatically and removes the need to trust a hardware manufacturer. For a two-factor solution the token is stored unencrypted on the device, and the PIN is memorised by its owner. This is at the cost to the server of protecting a single secret for use in signature verification.

As a full featured non-repudiable signature scheme the method would we suggest have application beyond simple authentication, in contexts where a multi-factor signature capability were desirable, as in our original banking scenario.

References

1. The FIDO alliance. `https://fidoalliance.org/`.
2. The international PGP home page. `http://www.pgpi.org/`.
3. S. Al-Riyami and K. Paterson. Certificateless public key cryptography. Cryptology ePrint Archive, Report 2003/126, 2003. `http://eprint.iacr.org/2003/126`.
4. L. Ballard, M. Green, B. de Medeiros, and F. Monrose. Correlation-resistant storage via keyword-searchable encryption. Cryptology ePrint Archive, Report 2005/417, 2005. `http://eprint.iacr.org/2005/417`.
5. M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. In *Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 268–286. Springer-Verlag, 2004.
6. J. Cha and J. Cheon. An identity-based signature from gap Diffie-Hellman groups. In *PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30. Springer-Verlag, 2003.
7. S. Chatterjee and A. Menezes. On cryptographic protocols employing asymetric pairings. *Discrete Applied Mathematics*, 159(13):1311–1322, 2011.
8. S. Galbraith, K. Paterson, and N. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156:3113–3121, 2008.
9. F. Hess. Efficient identity based signature schemes based on pairings. In *SAC 2002*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer-Verlag, 2003.
10. M. Scott. Authenticated ID-based key exchange and remote log-in with simple token and PIN number. Cryptology ePrint Archive, Report 2002/164, 2002. `http://eprint.iacr.org/2002/164`.
11. X. Yi. An identity-based signature scheme from the Weil pairing. *IEEE Communications Letters*, 7(2):76–78, 2003.

Time Permits

A Time Permit in its most common usage is simply a blob of data issued on request to a registered signer on the actual date D when they are asked to, or wish to, sign a document. It is obtained by summing components issued by each individual D-TA. Since the time permit is tied to an individuals pre-registered identity, it need not be protected in any way, and is of no value to an adversary. However without one, a valid signature cannot be formed. Therefore revocation is achieved by simply not issuing a Time Permit. Furthermore this date D now forms part of the signature, and so the verifier now has a cryptographic assurance of the date on which the signature was created.

We describe the mechanism for the basic scheme – the extension to the scheme with non-repudiation is straightforward. The public key for Alice is now formed by Alice as $A = H(ID_a) + H_2(D|ID_a)$ where $H_2(.)$ is a hash function distinct from, but with the same range as, $H(.)$. The time permit issued for Alice would be of the form $T = sH_2(D|ID_a)$, which must be added to Alices secret key sA. So the signature on a message m is now the tuple $\{ID_a, D, U, V\}$, where $U = xA$, and $V = -(x + h(m, U))(sA + T)$. The verifier uses the claimed date from the signature and the identity string ID_a to construct Alice's public key.