Distributed "Magic Ink" Signatures

Markus Jakobsson* Moti Yung †

Abstract

The physical analog of "blind signatures" of Chaum is a document and a carbon paper put into an envelope, allowing the signer to transfer his signature onto the document by signing on the envelope, and without opening it. Only the receiver can present the signed document while the signer cannot "unblind" its signature and get the document signed.

When an authority signs "access tokens", "electronic coins", "credentials" or "passports", it makes sense to assume that whereas the users can typically enjoy the disassociation of the blindly signed token and the token itself (i.e. anonymity and privacy), there may be cases which require "unblinding" of a signature by the signing authority itself (to establish what is known as "audit trail" and to "revoke anonymity" in case of criminal activity).

This leads us to consider a new notion of signature with the following physical parallel: The signer places a piece of paper with a carbon paper on top in an envelope as before (but the document on the paper is not yet written). The receiver then writes the document on the envelope using *magic ink*, e.g., ink that is only visible after being "developed". Due to the carbon copy, this results in the document being written in visible ink on the internal paper. Then, the signer signs the envelope (so its signature on the document is made available). The receiver gets the internal paper and the signer retains the envelope with the magic ink copy. Should the signer need to unblind the document, he can develop the magic ink and get the document copy on the envelope. Note that the signing is not blinded forever to the signer. We call this new type of signature a *magic ink signature*.

We present an efficient method for distributively generating magic ink signatures, requiring a quorum of servers to produce a signature and a (possibly different) quorum to unblind a signature. The scheme is robust, and the unblinding is guaranteed to work even if a set of up to a threshold of signers refuses to cooperate, or actively cheats during either the signing or the unblinding protocol. We base our specific implementation on the DSS algorithm. Our construction demonstrates the extended power of distributed signing.

*Department of Computer Science and Engineering, University of California, San Diego. markus@cs.ucsd.edu

†CertCo, New York, NY. moti@cs.columbia.edu, moti@certco.com

1 Introduction

In recent years, various notions of distribution of cryptographic functions (signature and encryption) among independent agents were considered. The typical added functionality of such a distribution include increased security of the secret key, increased availability of service, and increased flexibility of access, the latter by requiring a quorum to access information (as in, e.g., [9, 12, 20]). All these notions are functionality of distributed computing.

In this work we suggest that the distributed signature setting also provides for extended functionality by enabling "a new notion of signature itself" which is otherwise impossible (owing to the added control in this case). The notion we specifically suggest is that of "Magic Ink Signatures". In such a signature service, the signer blindly signs a message for a receiver, while retaining the capability to "unblind" the signature (analogous to developing "Magic Ink"), at any later point. What the distribution enables us is to implement the unblinding with separation in time – i.e., allowing the development of the "Magic Ink" at some point, but not earlier. This is impossible in the centralized case (what the signer can do at some point it can do earlier if there is no limiting factor such as the "Quorum Control" in the distributed case).

Note that requiring various actions of a quorum of distributed agents regarding a specific signature value needs a careful flexible design. For example, we cannot require that in each action the same identical quorum of agents be present. Requiring this may, paradoxically, reduce the availability of the service as the distribution level grows (whereas one of the initial reasons in distributing the service was increased availability). For the same reason, and quite counter-intuitively, it may also force us to put *more* trust in individual servers with a higher degree of distribution, unless care is taken.

The magic ink signature enables the generation of blind signatures which can later be unblinded by the signer (following the physical analogue given in the abstract). This is in sharp contrast with traditional blind signatures, which are information theoretically blinded to the signer [5]. The typical application where the need for unblinding arises is for cases where "privacy of individuals" is assured until some criminal or otherwise unusual activity is detected. Upon detection, identification of the origin of a signature becomes important in identifying the source of the unwanted activity. This is applied to private access tokens, authorized anonymous accounts, and electronic-money. Regarding the later setting, Chaum, Fiat and Naor's [7] original off-line scheme (and its follow-ups) offered perfect anonymity. However, the absolute privacy feature of all these schemes is not only beneficial to honest users, but also to criminal offenders, as it makes perfect crimes possible [2, 4, 8, 11, 18, 23, 25]: Various methods for anonymity revocation are suggested in some of these works mentioned. In "fair blind signatures" [3, 8, 11, 25], a signature receiver puts a pseudonym into the signature, allowing a third party (a judge) to later unblind the signature by calculating a pseudonym from a signature or vice versa. Magic

ink signatures are the "distributed cousins" of fair blind signatures, increasing the availability and lowering the amount of trust required (no need to employ a third party beyond the distributed signing agents and using quorum control to assure separation of duties). Magic ink signatures is a generic tool for blind signature generation, enabling the possibility of unblinding selected blind signatures by the signer – but only under quorum agreement to do so. The method is applied to a payment scheme with revocable privacy in [17, 19]. We note that it is easy to apply proactive methods [14, 15] to our suggested solution, for maximum security and availability.

Organization: We present a magic ink signature scheme that is robust, ensuring that as long as a quorum of (a plurality of honest) servers cooperate, they will *always* be able to unblind a given signature. Thus, we ensure availability and a high degree of distribution and reduced degree of trust required from an individual server. We first specify the notion of magic ink signatures, and the format of DSS signatures. Then, in section 3, we present the intuitive approach of magic ink DSS signatures. In section 4, we explain the model, our assumptions, and the tools we utilize. Among these is a new construction of robustness applicable to certain distributed protocols. This is followed by a protocol for magic ink generation of DSS signatures in section 5. In section 6, we elaborate on the robustness of the scheme and we claim its properties in section 7 and the Appendix.

2 Requirements and Background

Specifications: We wish to obtain a signature scheme where blind signatures can be distributively produced by a quorum of trustees, and these signatures can *always* be unblinded by a (possibly different) quorum (assuming a certain linear-fraction majority of honest trustees). We specify the following properties:

- Signatures are generated using a (t, n) threshold scheme by any t out of the n trustees. Less than t trustees cannot generate a valid signature.
- The signatures are computationally blinded to any set of less than t trustees (i.e., the signature cannot be correlated to the blinded signature or the signing session by a set of less than t trustees.)
- Valid signatures can be unblinded, i.e., signatures matched to signing session or vice versa, by any t out of the n trustees, regardless of the behavior of the other $n - t$ trustees and the signature receiver.
- Furthermore, we want signatures generated by an attacker who compromises the secret key of less than t signers (or forces these signers to sign using a protocol different than the specified) to be identifiable by any t of the signers (i.e., having an audit trail of legal signatures).

2.1 The Digital Signature Standard (DSS)

We use the DSS (described herein) as the underlying signature algorithm [21].

Note: Since we use different moduli at different times, we use $[op]_z$ to denote the operation op modulo z, where this is not clear from the context.

Key Generation. A DSS key is composed of public information p, q, g, a public key y and a secret key x, where:
1. p is a prime number of length l where l is a multiple of 64 and $512 \leq l \leq 1024$.
2. q is a 160-bit prime divisor of $p-1$.
3. g is an element of order q in Z_p^*. The triple (p, q, g) is public.
4. x is the secret key of the signer, a random number $1 \leq x < q$.
5. $y = [g^x]_p$ is the public verification key.

Signature Algorithm. Let $m \in Z_q$ be a hash of the message to be signed. The signer picks a random number k such that $1 \leq k < q$, calculates $k^{-1} \bmod q$ (w.l.o.g. k and k^{-1} values compared to DSA description are interchanged), and sets

$$\begin{aligned} r &= [[g^{k^{-1}}]_p]_q \\ s &= [k(m + xr)]_q \end{aligned}$$

The pair (r, s) is a signature of m.

Verification Algorithm. A signature (r, s) of a message m can be publicly verified by checking that $r = [[g^{ms^{-1}} y^{rs^{-1}}]_p]_q$.

3 Single-Server (Pseudo) Magic Ink Signatures

In order to communicate the intuition of our scheme, we present a method for producing Magic Ink DSS Signatures using only one signing server (which will be able to unblind the signature at will at any time). However, when we later distribute the signature server, signing and unblinding both will require quorum agreement.

3.1 (Pseudo) magic ink generation of DSS signatures

1. The signature receiver R has a hashed message $m \in Z_q$ that he wants signed. He generates two blinding factors, $a, b \in_u Z_q$, and computes a blinding of m, $\mu = [ma]_q$. R sends μ to the signature generating server S.

2. S generates a random secret session key, $\overline{k} \in_u Z_q$, and computes $\overline{r} = [g^{\overline{k}^{-1}}]_p$, which is sent to the signature receiver R.

3. The signature receiver R computes $r = [[\overline{r}^b]_p]_q$, and computes a blinding ρ of r: $\rho = [ra]_q$. R sends ρ to the signature generating server.

4. S generates a tag tag and the DSS signature σ on the message μ, using the public session key ρ. Here, tag is calculated first (which we describe how to do below), after which σ is calculated as follows: $\sigma = [\overline{k}(\mu + x\rho)]_q$. The server sends σ to R.

5. The signature receiver R unblinds the signature: $s = [\sigma a^{-1} b^{-1}]_q$. The triple (m, r, s) is a valid DSS signature on m.

Theorem 1: The protocol produces correct DSS signatures.

Proof of Theorem 1:
Recall that $\sigma \equiv_q \overline{k}(\mu + x\rho)$, $s \equiv_q \sigma a^{-1} b^{-1}$, $m \equiv_q \mu a^{-1}$, and $y \equiv_p g^x$. We can describe r either as $r = [[g^{\overline{k}^{-1} b}]_p]_q$ (from the point of view of information going from the signer(s) to the receiver) or as $r = [\rho a^{-1}]_q$ (from the point of view of information going back from the receiver to the signer(s)). We have that $[g^{m s^{-1}} y^{r s^{-1}}]_p = [g^{(m + xr) s^{-1}}]_p = [g^{(\mu a^{-1} + x\rho a^{-1}) s^{-1}}]_p = [g^{(\mu a^{-1} + x\rho a^{-1})(\sigma a^{-1} b^{-1})^{-1}}]_p$ $= [g^{a^{-1}(\mu + x\rho)((\overline{k}(\mu + x\rho)) a^{-1} b^{-1})^{-1}}]_p = [g^{a^{-1}(\mu + x\rho)(\overline{k}(\mu + x\rho))^{-1} ab}]_p = [g^{\overline{k}^{-1} b}]_p \equiv_q r$
Thus, the protocol generates valid DSS signatures. □

3.2 Generation of tags

Let us start by making the following observation:

Signature-View Invariant: Let $[mr^{-1}]_q$ identify a valid signature (m, r, s), and $\{\mu, \rho\}$ part of the the view of the signer during a signature generation session. We have: $mr^{-1} \equiv_q \mu\rho^{-1}$, since $\mu = [ma]_q$ and $\rho = [ra]_q$ for a valid signature.

Justification: We have $\sigma = [\overline{k}(\mu + x\rho)]_q$ for (μ, ρ) generated by R. Linear combinations of more than one such signature are not known to give a signature of the valid form (due to the use of different values of $\overline{k}$ of different signatures; and implied by our assumption of existential unforgeability of this type of signatures) so we will only consider operations on *one* signature. Multiplying the value σ by a coefficient can maintain a valid signature; two such manipulations are known: First, for $s = [\sigma a]_q$, and $(m, r) = ([\mu a]_q, [\rho a]_q)$ we have that (m, r, s) is still a valid signature. Second, for $s = [\sigma b]_q$, and $(m, r) = (\mu, \rho) = (\mu, [[g^{\overline{k}^{-1} b}]_p]_q)$, we have that (m, r, s) is also a valid signature. For both of these manipulations, the invariant holds, and no other applicable blinding methods are known. Therefore, any way of obtaining a valid signature (m, r, s) for which $m/r \not\equiv_q \mu/\rho$, would give a new method for blinding of signatures of this type.

Use for tagging: We will use the signature-view invariant for the production of tags, which will be (possibly distributedly stored results of) a function of $[\mu\rho^{-1}]_q$. Consider the following tagging method: The signature servers distributively generate and keep a marker (session tag) specific to the signing session, and a distributed tag (unknown to any subset of less than t servers). Together, these can be used to distributively calculate the invariant $[m/r]_q$ of the related session, which can be output and compared to a signature invariant (based on m and r) or distributively (secretly) compared to a given invariant.

3.3 Tracing

There are three types of tracing we can perform:

1. **From known signing session to signed message:** The signature invariant is calculated from the tag and the marker of a given session.
2. **From known signed message to signing session:** The given signature invariant is distributively compared to the signature invariant of each potential session, which is distributively calculated from the tag and the marker of a given session.
3. **By comparison:** The given signature invariant is distributively compared to the signature invariant of the given session.

4 Model and Tools

4.1 Communication and Threat Model

We assume the standard computational model of polynomial-time randomized Turing machines. Players are connected by an insecure broadcast medium, and an (also polynomial time limited) adversary can inject messages and eavesdrop, but not disconnect any other player from the network. Furthermore, the adversary can corrupt up to $t-1$ of the n players in the network, and by doing so, force the corrupted players to divert from the specified protocol arbitrarily. See [12] for more details about the model.

4.2 Assumptions

We will rely on the following assumptions:

1. The Undeniable Signature Assumption [6] holds (i.e., given an input quadruple (m, s, g, y), it is hard to decide whether $log_m s = log_g y$, unless $x = log_g y$ is known.) This implies that the Discrete Log problem is not in BPP, and that Pedersen's secret sharing scheme [22] is secure on random secrets.

2. The DSS signature scheme where the signature receiver is allowed to specify the message m to be signed *after* seeing the value $[g^{k^{-1}}]_p$ is secure against a chosen message attack.

4.3 Tools

Let us briefly describe the existing tools we employ:

- **Polynomial Interpolation Secret Sharing**[24]: This is the well-known result in which a secret σ is shared by choosing at random a polynomial $f(x)$ of degree t, such that $f(0) = \sigma$.
- **Joint Random Secret Sharing**[10, 22]: In a Joint Random Secret Sharing scheme the players *collectively* choose shares corresponding to a (t, n)-secret sharing of a random value.
- **Joint Zero Secret Sharing**[1]: This protocol generates a *collective* sharing of a "secret" whose value is zero. Such a protocol is similar to the above joint random secret sharing protocol but instead of local random secrets each player deals a sharing of the value zero.
- **Computing Reciprocals**[12]: Given a secret $k \bmod q$ which is shared among players $P_1, ...P_n$, generate a sharing of the value $k^{-1} \bmod q$, without revealing information on k and k^{-1}.
- **Multiplication of Secrets**[12]: Given two secrets u and v, which are both shared among the players, compute the product uv, while maintaining both of the original values secret (aside from the obvious information which is revealed from the result).

The multiplication of two secrets easily extends to linear combinations and products of three secrets, e.g., $\overline{k}_i(\mu_i + x_i\rho_i)$ for secrets $\overline{k}_i$, μ_i, x_i, and ρ_i. This is achieved without altering the method given in [12]. We also use three new tools:

- **Comparison of Secrets**: Given two secrets u and v, which are both shared among the players (or one is shared one is known), using the above tools we can compare their equality without learning the secret values.
- **Undeniable Signature Based Robustness**: We introduce the use of the verification protocol of undeniable signatures to prove correct exponentiations.
- **Destructive Robustness**: We introduce a new method for making distributed protocols robust: Instead of verifying that each individual share of the calculation is correct, we first combine the shares and then verify

that the combined result is correct. If it is not, then each share of the result is verified. A minor efficiency improvement is obtained from doing so. But more importantly, this approach allows simpler and clearer protocol design. This is because we can allow the individual correctness verification to destruct important properties of the produced transcript, which, if the combined result is not correct, is a worthless transcript anyway. Therefore, we call this type of robustness *destructive robustness.*

5 Magic Ink Signature Generation

5.1 Distributed magic ink generation of DSS signatures

Let us now consider a distributed version of the protocols previously presented. Here, let Q be a quorum of t servers in $S_1 \ldots S_n$:

1. The signature receiver R has a message $m \in Z_q$ that he wants signed. He generates two blinding factors, $a, b \in_u Z_q$. He then computes a blinding of m, $\mu = [ma]_q$, and a (t, n) secret sharing $(\mu_1, \ldots \mu_n)$ of μ, with public information $(g^{\mu_1} \ldots g^{\mu_n})$. He sends μ_i to signature generating server S_i.

2. The set of servers $S_i | i \in Q$ distributively generate a random secret session key, $\overline{k} \in_u Z_q$, where server S_i has a share $\overline{k}_i$. Server S_i publishes $[g^{\overline{k}_i}]_p$, and using the methods for computing reciprocals in [12], the servers compute $\overline{r} = [g^{\overline{k}^{-1}}]_p$, which is sent exclusively to the signature receiver R.

3. The signature receiver R computes $r = [[\overline{r}^b]_p]_q$, and blinds this: $\rho = [ra]_q$. R computes a (t, n) secret sharing $(\rho_1, \ldots \rho_n)$ of ρ, with public information $(g^{\rho_1} \ldots g^{\rho_n})$. R sends ρ_i to S_i.

4. The set of servers $S_i | i \in Q$ distributively generate the tag tag and the DSS signature σ on the message μ, using the (shared) public session key ρ. Here, tag is calculated first (for which we present a robust protocol below), after which σ is calculated as follows: S_i generates $\sigma_i = [\overline{k}_i(\mu_i + x_i\rho_i)]_q$. Then, $\sigma = [\overline{k}(\mu + x\rho)]_q$ is interpolated from the σ_i's using the method for multiplication of secrets in [12]. The servers send σ to R.

5. The signature receiver R unblinds the signature: $s = [\sigma a^{-1} b^{-1}]_q$. The triple (m, r, s) is a valid DSS signature on m.

We note that the proof of correctness is identical to that of the non-distributed protocol version, given robust primitives for secret sharing (e.g., [10, 22]), for computing reciprocals (e.g., [12]) and for multiplication of secrets (e.g., [12]). Also note that we can use standard zero-knowledge techniques to force the receiver to prove that the blinding of steps 1 and 3 are consistent.

5.2 Distributed tag generation and tracing

Let us review the steps of tagging method previously outlined: At the time of signing, μ and ρ are available distributedly. The servers distributively compute $[\mu/\rho]_q$ (without revealing this value to each other). Also, they select a distributed random value $[c]_q$. The servers distributively store this value, and its inverse $[c^{-1}]_q$. They compute and publish the tag $[c(\mu/\rho)]_q$ (given that the components of the multiplication are distributed and secret, this value is random). We can now trace from a session to a signature by distributed multiplication of the tag by $[c^{-1}]_q$, and comparing the result to the public signature invariant. Given a signature invariant $[m/r]_q$, we can distributedly multiply by a value $[c]_q$; if the (distributively held) result equals the the published tag, the session and the signature indeed match. For comparison of a session to a signature, on the other hand, we do not reveal the result of the last multiplication. Rather we check distributedly and secretly for equality of the computed multiplication and the session tag. Note that the probability of collision of tags is negligible.

6 Robustness of Signature Generation

So far, we have not considered the robustness of the signature generation. We will employ *destructive robustness* in order to obtain high efficiency without sacrificing anonymity.

Destructive robustness involves two steps: (1) combination of shares of the result, and error detection, by verifying the correctness of the combined result. This check can be done either internally (i.e., by the same entities that produced the shares) or externally. Then, if the combined result is not correct, the second step is invoked: (2) error tracing, in which it is determined which server(s) have deviated from the protocol. This kind of robustness is possible in protocols where partial incorrect results can be discarded and when we can withstand delays of malicious servers revealing themselves in a slow pace.

We demonstrate an *external* method of destructive robustness for the generation of the blind signature σ on μ, using ρ as public session key:

1. **Share Combination and (External) Error Detection:**
   The signature servers send σ to R, who unblinds the result, obtaining a triple (m, r, s). If this signature is not valid, then R sends a complaint to the signature servers, invoking the next step:

2. **Error Tracing:**
   $S_i | i \in Q$ reveals μ_i. If σ_i was computed correctly, then $g^{\sigma_i} \equiv_p g^{\overline{k}_i(\mu_i + x_i\rho_i)} \equiv_p ((g^{\mu_i})g^{x_i\rho_i})^{\overline{k}_i} \equiv_p ((g^{\mu_i})y_i^{\rho_i})^{\overline{k}_i}$. Using a verification protocol for undeniable signatures, S_i proves that for some $I = (g^{\mu_i})y_i^{\rho_i}$ it is true that $log_I g^{\sigma_i} = log_g(g^{\overline{k}_i})$. He then proves that $log_{y_i}(I(g^{\mu_i})^{-1}) = log_g(g^{\rho_i})$. A server S_i is declared a cheater if he refuses to reveal the information, if the

information is not consistent with the public shares of the secret sharing schemes, or if the share s_i sent out earlier was incorrectly computed.

We see that the above method assures that cheating servers are caught, and that no transcript properties are lost when no complaint is filed. Also note that if R files a unjustified complaint, then *this* will be established, since it will be found that no server cheated. Finally, note that no secret information of honest servers will be leaked to R if R receives an invalid signature transcript. R has no motivation to complain about a good signature; this results in early "unblinding". Each time a threshold is used and opened, the misbehaving processors are eliminated and the process start afresh (to avoid leaking information) based on new random choices. This may result in a delay of at most t times, but enables t to be a maximal minority $n = t/2 + 1$. Note that the method is applicable due to the probabilistic nature of the computation and the care in opening erroneous results.

7 Correctness Claims

We claim that the scheme satisfies the specification of Magic Ink Signature schemes. More specifically, we claim that

- We generate correct DSS signatures in a robust way, using a (t, n) threshold scheme (t from [12]).
- It is not possible for less than t out of n signature servers to correlate a signed message to its blinded withdrawal session.
- It is always possible for t out of n signature servers to correlate a signed message to its blinded withdrawal session.
- It is always possible for t out of n signature servers to distinguish messages they signed from messages signed by an attacker who compromised their secret key (or forced them to produce a signature in a fully blinded manner.)

The claims are shown to hold in the appendix.

Finally, we note that key exchange can be reduced to magic-ink signatures (one party playing the receiver and the other party plays all signers, and the message being the key). Due to blinding, the message (key) is hidden from eavesdroppers, but not from the party playing the signers (since it can perform "unblinding" internally). This implies the difficulty of designing magic-ink signature merely based on the existence of a general one-way permutations [16].

8 Acknowledgments

Thanks to Russell Impagliazzo for numerous discussions, to Jan Camenisch and Markus Stadler for pointing out corrections to the initial draft, and to Markus Michels, Tal Rabin and Rebecca Wright for helpful comments and remarks.

References

[1] M. Ben-Or, S. Goldwasser, A. Wigderson, "Completeness Theorems for Non-cryptographic Fault-Tolerant Distributed Computations," STOC '88, pp. 1-10.

[2] E. Brickell, P. Gemmell, D. Kravitz, "Trustee-based Tracing Extensions to Anonymous Cash and the Making of Anonymous Change," Proc. 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 1995, pp. 457-466.

[3] J. Camenisch, U. Maurer, M. Stadler, "Digital Payment Systems with Passive Anonymity-Revoking Trustees," Computer Security - ESORICS 96, volume 1146, pp. 33-43.

[4] J. Camenisch, J-M. Piveteau, M. Stadler, "An Efficient Fair Payment System," 3rd ACM Conf. on Comp. and Comm. Security, 1996, pp. 88-94.

[5] D. Chaum, "Blind Signatures for Untraceable Payments," Advances in Cryptology - Proceedings of Crypto '82, 1983, pp. 199-203.

[6] D. Chaum, H. Van Antwerpen, "Undeniable Signatures," Advances in Cryptology - Proceedings of Crypto '89, pp. 212-216.

[7] D. Chaum, A. Fiat and M. Naor, "Untraceable Electronic Cash," Advances in Cryptology - Proceedings of Crypto '88, pp. 319-327.

[8] G.I. Davida, Y. Frankel, Y. Tsiounis, and M. Yung, "Anonymity Control in E-Cash Systems," Financial Cryptography 97.

[9] Y. Desmedt, Y. Frankel, "Threshold Cryptosystems," Advances in Cryptology - Proceedings of Crypto '89.

[10] P. Feldman, "A Practical Scheme for Non-Interactive Verifiable Secret Sharing" FOCS '87, pp. 427-437.

[11] Y. Frankel, Y. Tsiounis, and M. Yung, "Indirect Discourse Proofs: Achieving Efficient Fair Off-Line E-Cash," Advances in Cryptology - Proceedings of Asiacrypt 96, pp. 286-300.

[12] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, "Robust Threshold DSS Signatures", Advances in Cryptology - Proceedings of Eurocrypt '96, pp. 354-371.

[13] S. Goldwasser and S. Micali, "Probabilistic Encryption". J. Comp. Sys. Sci. 28, pp 270-299, 1984.

[14] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, M. Yung, "Proactive Public Key and Signature Systems," 4th ACM Conf. on Comp. and Comm. Security, 1997.

[15] A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung, "Proactive Secret Sharing, or How to Cope with Perpetual Leakage," Advances in Cryptology - Proceedings of Crypto '95.

[16] R. Impagliazzo and S. Rudich, Limits on the Provable Consequences of One-way Permutations, STOC '89.

[17] M. Jakobsson, "Privacy vs. Authenticity," PhD Thesis, University of California, San Diego, Department of Computer Science and Engineering, 1997. Available at `http://www-cse.ucsd.edu/users/markus/`.

[18] M. Jakobsson and M. Yung, "Revocable and Versatile Electronic Money," 3rd ACM Conference on Comp. and Comm. Security, 1996, pp. 76-87.

[19] M. Jakobsson and M. Yung, "Applying Anti-Trust Policies to Increase Trust in a Versatile E-Money System," Financial Cryptography '97.

[20] S. Micali, "Fair Cryptosystems," Advances in Cryptology - Proceedings of Crypto '92.

[21] National Institute for Standards and Technology, "Digital Signature Standard (DSS)," Federal Register Vol 56(169), Aug 30, 1991.

[22] T.P. Pedersen, "Distributed Provers with Applications to Undeniable Signatures," Advances in Cryptology - Proceedings of Eurocrypt '91, pp. 221-242.

[23] S. von Solms and D. Naccache, "On Blind Signatures and Perfect Crimes," Computers and Security, 11 (1992) pp. 581-583.

[24] A. Shamir, "How to Share a Secret," CACM, V. 22, 1979, pp. 612-613.

[25] M. Stadler, J-M. Piveteau, J. Camenisch, "Fair Blind Signatures," Advances in Cryptology - Proceedings of Eurocrypt '95, 1995.

9 Appendix: Correctness and Security

The magic ink signature generation is correct, as shown in the proof of the single-server version (which generates the signature the same way) in section 3, and its simulation in the distributed setting (based on [12]). The robustness of the signature generation depends on our destructive robustness method for random signatures (on top of the non-robust threshold DSS), and the soundness of the composed undeniable signature verification.

Let us next sketch the proof of the additional required properties: that the original signature is blinded and that it can be unblinded as well.

Theorem 2: A coalition of less than t cheating servers cannot, with a non-negligible advantage over guessing, correlate a signature to a signing session.

Proof of Theorem 2: (Sketch)
Let V_1 and V_2 be the view of a coalition of less than t signing servers for two different signing sessions. Let (m, r, s) be a signed message, created in either one of these signing sessions, and assume, in order to reach a contradiction, that the signed message can be correctly matched to either V_1 or V_2 with probability $\frac{1}{2} + \varepsilon$, where ε is polynomial in the size of the security parameters.
We will show that this is not possible by demonstrating that, unless the undeniable signatures assumption is invalid, a polynomial time limited adversary will not be able to tell the transcript parts given from random strings.
We will therefore perform the following thought experiment: We assume that we have a random string of the size of the signers' view during a signature generation phase, where each individual part of the string (corresponding to a transcript part (communication step) of the generation) is selected from the same distribution as its corresponding actual transcript part. Then, we will replace the individual parts of the random string with part of a real transcript one by one. For each step, we will show that it is not possible for a non-quorum of signers to distinguish which of the strings corresponds to a string before or after the last replacement. (This is the walking argument on a random variable [13]). This shows that given a generated signature, its generation view and a random string of the same size and same public distribution, a non-quorum of signers will not be able to match the signature to the the generation view with more than a negligible probability.
We divide the information into two sets, (a) the view of less than a threshold of signer servers, and (b) the signed message. The view of server S_i consists of the public information $\{g^{\mu_i}\}$, $\{g^{\overline{k}_i}\}$, $\overline{r} = [g^{\overline{k}^{-1}}]$, $\{g^{\rho_i}\}$ $\{\sigma_i\}$, σ. We also have the value tag, and intermediary results in the generation; we will consider this later. We have private information $\overline{k}_i$, μ_i, ρ_i and x_i. Since the private information are random shares of $\overline{k}$, μ, ρ and x; and σ_i is just a combination of the random shares and public information, these (or less than t of these sets) cannot help us to correlate the view to the signed message. Therefore, we focus only on

the public information and the signed message, and prove that these cannot be correlated by a non-threshold of signature servers.
Let us consider what meaningful information can be calculated from the public view and the signed message: The signed message is of the form (m, r, s), where $r = g^{\overline{k}^{-1}b}$ and s is such that $r \equiv_q g^{ms^{-1}} y^{rs^{-1}}$. Given the structure of the tag, we need to learn something about $[\mu/\rho]_q$ in order to trace (which we will show to be hard). Without loss of generality, let us consider real transcripts parts of V_1, and the following order of substituting correct transcript parts with random transcript parts in the list of random transcript parts and a potential triple (m, r, s). The following are ideas regarding the implications of possible distinguishability at each substitution stage:

1. Substitute in g^{μ}: It is not possible to distinguish this step, since (m, r, s) is statistically uncorrelated to μ (given that a is chosen uniformly at random, $\mu = [ma]_q$, r is not related to μ, and s is uncorrelated from μ by b, which is chosen uniformly at random.)

2. Substitute in $g^{\overline{k}}, \overline{r} = g^{\overline{k}^{-1}}$: It is not possible to distinguish this step either, since (m, r, s, μ) is statistically uncorrelated to $\overline{k}$. It cannot be correlated to m or μ since these are not related, and not to r since b is chosen uniformly at random and $r = [[g^{\overline{k}^{-1}b}]_p]_q$. It cannot be correlated to s, which is a linear combination of $\overline{k}$, x (both unknown) and μ, r (both in the set of potential transcript parts), or: given the linear combination, and known μ,$\overline{r}$,σ,ρ we would be able to decide the undeniable signature $(g, g^x, \overline{r}, \overline{r}^x)$, where $\overline{r}^x = (g^{\sigma}\overline{r}^{-\mu})^{1/\rho}$. The same argument holds for substituting in $g^{\overline{k}^{-1}}$.

3. Substitute in g^{ρ}: We see that ρ is unrelated to m, and $g^{\overline{k}}$. If we can correlate it to r or $\overline{r}$, this gives us an algorithm for deciding the undeniable signature (g, g^b, r, r^b) (assuming a is known); s is just a linear combination of the above, and the previous argument for linear combinations holds. It is not possible to produce a known function $[\mu/\rho]_q$ from g^{μ} and g^{ρ}, or the Diffie-Hellman assumption breaks.

4. Substitute in σ_i (or with the same argument: σ): Follows the linear combination argument above.

Next, based on the above ideas, since none of the substitutions can be distinguished from a random string, it is not possible to match with related non-negligible probability one signed message to one out of two signing views (by the triangle inequality). Let us consider *tag* now: The tag generation protocol outlined is specified so that it hides the participants inputs (guaranteed by the properties of the protocol for multiplication of secrets and inverting a secret.) In fact the public tag is a random element mod q (for each tag and signature

invariant there is an element that matches it with the signature). It is therefore not possible to match a signed message to one out of two possible signing views.

It is also true, then, that it is not possible to match a signed message to one out of n signing views. Otherwise we simply would get a contradiction by constructing $n-2$ additional signing views, none of which matches the signed message, and then match this to one of the remaining two views. □

Theorem 3: A quorum of t servers will always succeed in unblinding a signature in either of the three directions given.

Proof of Theorem 3: (Sketch)
Given that $[m/r]_q \equiv_q [\mu/\rho]$ is always true for a signature generation session in which a valid signature (m, r, s) is generated (this can be guaranteed using zero-knowledge proofs if a new blinding methods is suspected to exist), we have that the tag will always be retrievable given robust protocols for tag generation and tracing. The robustness of these follows from the robustness of methods for multiplication and inversion of secrets. □

An audit trail of legal signatures:
In addition to correct checking and tracing of existing signatures, we have a built-in fraud detection mechanism. Since only signatures that were generated in the proper manner by the signature servers will have a tag, it will be possible to distinguish such signatures from signatures generated by an attacker who compromised the secret key of the signer but has no access to the tags. The signature servers can compare tag by tag to the signature (using the third tracing option,) and if no tag matches, then it is invalid. This feature may provide an "audit trail" for sensitive services.