I Did Not Accept That: Demonstrating Consent in Online Collection of Personal Data

Vitor Jesus and Shweta Mustare

School of Computing and Digital Technology, Birmingham City University, Birmingham, United Kingdom
vitor.jesus@bcu.ac.uk

Abstract. Privacy in online collection of personal data is currently a much debated topic considering, amongst other reasons, the incidents with well known digital organisations, such as social networks and, in Europe, the recent EU/GDPR regulation. Among other required practices, explicit and simply worded consent from individuals must be obtained before collecting and using personal information. Further, individuals must also be given detailed information about what, how and what for data is collected. Consent is typically obtained at the collection point and, at a single point in time (ignoring updates), associated with Privacy Policies or End-User Agreements. At any moment, both the user and the organization should be able to produce evidence of this consent. This proof should not be disputable which leads us to strong cryptographic properties.
The problem we discuss is how to robustly demonstrate such consent was given. We adapt fair-exchange protocols to this particular problem and, upon an exchange of personal data, we are able to produce a cryptographic receipt of acceptance that any party can use to prove consent and elicit non-repudiation. We discuss two broad strategies: a pure peer-to-peer scheme and the use of a Trusted Third Party.

Keywords: Privacy, Fair Exchange, Consent

1 Introduction

Online privacy has always been a challenging problem that, nevertheless, has been exacerbated with technological advances in data processing and the proliferation of sensors and mobile technologies. Individuals find hard to not only track the flows of personal data but also to establish exactly what organisations (the Data Controllers or Processors) are collecting and possess about them. Furthermore, the conditions in which consent was given are also, more often than not, unclear. The EU General Data Protection Regulation (EU/GDPR), effective in May 2018, is seen as an advance in the control individuals have over their personal data and covers European citizens. Recently, multiple well-known digital organisations have also been under the spotlight for collecting and sharing information in ways that, at best, are unclear to individuals. This motivates our work.

The central concept at stake is *consent*. Even if not sufficient in many cases, it is consensually necessary before personal information is collected. Among others, a key requirement of EU/GDPR is that individuals need to explicitly consent with the treatment and purpose of the collected personal data (i.e. opt-in). This further includes revealing which third parties (the Data Processors) will have access to the personal data. Communication with the user must also be clear and simple. The organisation needs to put effort in simplifying how it is communicated to the user, e.g., when explaining their Privacy Policy or Terms of Agreement.

The problem we tackle in this paper is the following: when a User sends personal data under a Privacy Policy, and after a user gives consent, and in case of dispute, how can both the online Service and the User prove (i) what data was collected, (ii) how it was collected and (iii) the terms that both parties consented with at the time? Note the problem is not only on the User side, e.g:

– a dishonest User claims to never have given consent to a particular set of Terms or Privacy Policy or even to never have given Personal Data at all, thus accusing the Service of abuse or theft; or

– a dishonest online Service claims the User consented with extra conditions, i.e., agreed to a specific Privacy Policy which is different to the ones the User truly accepted.

Note that, nowadays, the common practice is to simply rely on local records (e.g. log files) managed by each party. In case of a dispute, they prove very little as these records can be easily manipulated.

The scenario we discuss in this paper is widely common: a User creates an account, using a web browser, with an online Service (e.g., a social network, a mobile application or an email provider) by sending personal data (such as identity, address, etc.) and the personal data is regulated by jointly agreed terms of service and privacy policy. After the User reviews policies and terms, and then accepts, personal data is sent over. Two key elements are involved at this stage about the Agreement: (i) it must be clear, detailed and simple enough for any average user to understand and (ii) it should not be specific to any user unless there is a good reason, i.e., everyone in the same conditions (say, at around the same time and location) should obtain the same Terms. We will further assume there will be Third Parties that are reviewing such Terms so that users already are aware of its major implications and, above all, get reviews when the Terms are updated.

The key problem we tackle is how a User or a Service can prove that, at some past point in time, there was consent regarding a specific agreement. The problem is akin to non-repudiation with two challenges: (i) the whole interaction and process must be protected and (ii) users must be allowed to be anonymous to any level of degree which means the scheme cannot rely on legal identity.

The main contribution we offer is a practical scheme for both parties to generate *non-repudiation of receipt* (not origin) and thus demonstrate if and how consent was obtained and offered. It has thus extreme current relevance. To the best of our knowledge, this is the first publication to discuss this practical and abundantly common aspect.

The reminder of this paper consists of the following. In section 2 we review the background of the technologies used. In section 3 we define the problem, its requirements and provide a threat model. In section 5 we propose two broad strategies (peer-to-peer and using a Trusted Third Party) and in section 5 we present an implementation and evaluate key metrics. Section 6 concludes our paper.

2 Background

To the best of our knowledge, this is the first time to specifically address the problem of proving consent when an exchange of personal data occurs when registering for online services. In this section we provide a background on online consent (largely non-technical), user identity on the Internet and fair-exchange protocols.

2.1 Online Consent

Consent is a legal concept and therefore difficult to capture with technology [1]. The familiar experience of clicking and accepting terms of service is what legal experts call *click contracts* or *web-wrap agreements* [2]. At least since the EU/GDPR came in force in May 2018, consent now has a tighter formulation especially regarding its validity. In a nutshell, the data collected, the purpose and means of collection require clarity and simplicity. In the United Kingdom, and one would expect in most countries if not all, implicit trust is the key element: the user *trusts* the service will deliver as promised in exchange for what the user is passing over such as personal data [3].

The problem we face here is how to obtain cryptographic evidence of non-repudiation of receipt and under conditions that were clear for both parties before the exchange happened. In other words, the problem we discuss is, once a dispute has arisen, how both User and Service can prove, beyond any doubt, (i) what data was shared, (ii) how it was (technically) collected and (iii) under which Terms of Agreement. For example, the Service will not be able to trick the user into accepting hidden conditions and, at a future moment, claiming those conditions were exposed to the user. The User will not also be able to claim to have received a different Privacy policy which has importance to organisations.

2.2 User Identity

The current Internet is largely identity-less. At best, an online identity (say, an account) can be linked to a real identity when legal documents are available; sometimes, it can only be linked to a payment method. Most often, there is no linkage at all, except contextual ones, and the account is only valid locally to the service. This creates problems when attempting to design a non-repudiation protocol as all that parties can rely on is transaction receipts.

If we had legal identities on the Internet, the identity problem could be trivial. Nowadays, many identity systems support Public Key Infrastructure and identity cards are even, physically, smart cards holding certificates and keys issued by a government, stored in secure hardware and under a standardised security programme. Such keys would, at first sight, resolve the problem if all parties were able to digitally sign transactions. However, even in that case, they may not be legally valid. As the techno-legal community has been debating, using such cryptographic material presents several challenges such as [4]:

- Revoking certificates is problematic. If a breach of a private key is detected, it would invalidate all past transactions from a forensics perspective because it will always be difficult to assert when the keys were breached.
- Timestamping, despite seemingly simple, requires a Time authority that has to be delegated from a government.
- Public keys, even in case of a breach, can never expire as they would invalidate past valid transactions.
- In most countries, digital certificates have no legal value and evidence can only be based on declarations.

The net effect is that plain old signatures, or *acts of will* [4] (such as clicking a button), is still the only way to lawfully prove acceptance of something [5] in most parts of the world. This is aligned with Privacy regulations when requiring acceptance of terms to be explicit.

Considering that such legal elaborations is outside the scope of our paper, and in order to keep our approach as generic as possible, we simply make no assumptions on the source of user identity. To this regard, the user is whoever will be running the protocols we present in this paper at the moment of exchange. All we require is that

- there are means to contact the User outside the real-time transaction for the duration of the agreement; and
- there is a globally unique identifier for the User, or a username compounded with a globally unique identifier for the service such as a Fully Qualified Domain Name (such as an email address).

2.3 Fair-Exchange Protocols

Fair-exchange protocols are such that parties who do not trust each other, but wish to exchange something electronically, mutually generate digital evidence that satisfies proof of non-repudiation of receipt [6]. To note that *fair non-repudiation* is probably more accurate for our scenario but we chose to use the more widespread term. In our case, we exchange Personal Data for access to a service and under certain Terms of Agreement. Fair-exchange protocols have found different applications such as delivery of software against payment, delivery of invoices [7], certified email delivery [8] or consumer card payments [9]. The problem also shares similarities with multi-party computation [10].

Proven approaches exist that cover, broadly, two key requirements: multi-party [11] and whether there is a need for a Trusted Third Party [13][14]. To note, however, that it seems to be an impossibility to achieve strong non-repudiation and fairness in the sense that all parties are at the same starting point and obtain what was intended at the end, if we require no involvement of a commonly trusted mediating agent, a Trusted Third Party (TTP) [15]. A simple analogy is that, when exchanging something (say, software for a payment) someone has to go first and so the party that goes second can always exit the protocol and the first party could hardly prove it sent anything. Approaches such as gradual exchange [7], where parties exchange shares iteratively, do not strictly meet the fairness requirement in conditions such as asymmetric computing power [11]. This problem is nowadays solved with escrow parties that act as a TTP.

3 Consent Model

In this section we formalise our problem, discuss what we mean by Consent (and Agreement) and present our Trust model.

3.1 Agreement Model

We consider the following common scenario see Figure 1. A User (the Individual) is creating an account online which requires personal data to be collected and sent to a Service.

The key components are

- *Private Data*, the personal data that the User is about to share, D
- *Policy or Terms*, the Terms of using the Service that the Service imposes, T_S
- *Collection Logic*, the mechanism by which data is collected, L, which can be as simple as a HTML form and a POST method or a JavaScript script
- *Receipt*, a (cryptography-based) receipt proving the exchange of data and its context, R

Together, these elements create what we call an Agreement $A = (D, T_S, L)$ for which evidence of acceptance is receipt $R^t_A(U)$ generated at time t for user U. The generation and handling of the receipt is the central contribution of this paper.

Furthermore, we include in the model a Trusted Third Party (TP) that is optional. The TP has three functions:

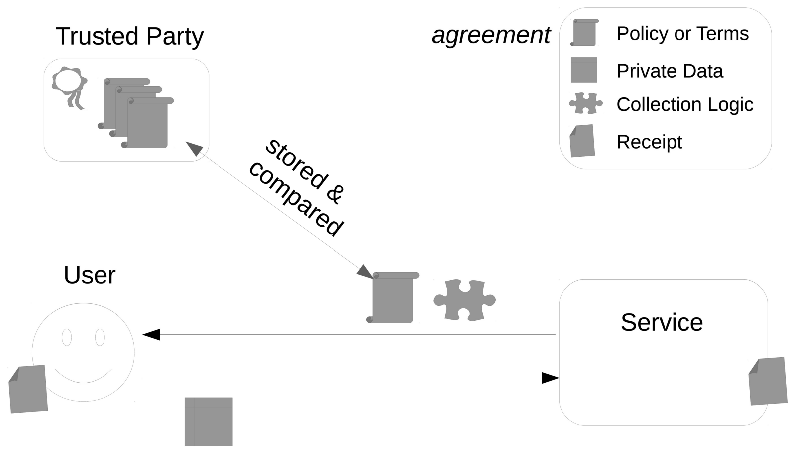


Fig. 1. Agreement model.

- First and most important, it serves as a trusted medium when neither user of service trust each other and neither wish to be in an unfair position such as sending first something. In other words, and in simple terms, each sends their contributions to the TTP and only when both completed their parts of the protocol the TTP releases to each the desired contribution. In practice, we will use an offline TTP which is only used in case a party exits the protocol prematurely. This will be detailed later.

- In a real-world scenario[1], a TTP publicly reviews (and possibly more) the Terms offered by the Data Controller to all equivalent users. We assume users, in practice, will not review in detail all the Privacy Policies and Terms of Service they come across. No matter how simplified and well communicated they might be, we assume users will mainly use reputation mechanisms and external reviews to fully understand the implications of how their personal data is used [12]and, if needed, to revoke at a later time the given consent, especially upon an update of the terms. Note that revoking consent will use the same mechanisms as described in this paper. Such entities already exist nowadays in some form (even if specialised news websites[2]) but we envision they will become more important with national regulations that impose clarity requirements and force online services to open they practices

[1] We are working with Privacy organisations towards piloting the ideas on this paper in a real-world scenario.
[2] See https://tldrlegal.com/ for software licences.

in more detail for external scrutiny.

– It has the capability of monitoring and recording the generic Terms of a given online Service; it may, further, as in one of our strategies, store the offered Terms to a particular user and can later be checked whether the user was offered the same conditions as any other user in the same conditions. This will mitigate the risk that the Service cheats by crafting a special Agreement at a given moment for a particular user thus misleading the user into accepting different Terms, for example, by manipulating the Terms are shown to the user revisit this aspect – see next section.

3.2 Collection Logic

We here make a simplification that, nevertheless, still defines an overwhelming common problem: we will assume the negotiation between user and service is conducted online and over common Web technologies. In particular, we assume the user interface is implemented in HTML and Javascript and the communication protocol is HTTP. This greatly helps for two reasons:

– *HTTP is stateless.* This means that capturing outgoing and incoming data is not dependent on any previous state. Very often the overall application is stateful as the back-end application servers store information which creates a session context. However, we assume that, upon exchanging personal data, the communication is truly stateless – even if it is designed for the sake of clarity. The result is that one should be able to completely capture the negotiation and the agreement and each end can save it for future records.

– *Implementation code is open.* We simply assume, as is virtually any case today, that the collection logic is based on Javascript (so running on a web browser) which, by nature, is completely auditable by the user as javascript arrives in source-code and is available for inspection. This means that, the combination of stateless HTTP and open source-code, together with the data (personal data and, say a privacy policy in the form of a file) provides a complete record of the transaction.

It should be noted that this is not the case for, for example, mobile apps. To give a practical example, the mobile application can intercept and modify, without the user knowledge, information (such as tricking the user to accept invisible terms); further, because it is not always possible to completely reverse-engineer or decompile a mobile application, one is never sure of what the actual collection logic was which creates room for disputes between the user and the service. The mobile app case, and generalising to any collection logic, is open for future work.

3.3 Trust Model

The problem we discuss is mutual: both User and Service need to prove the agreement they claim has been accepted. At any point in the future, both the User and Service can dispute what was agreed. The following lists the key threats and assumptions in our Trust model:

– *Service tricks the user into accepting non-generic Terms.* In other words, the Terms sent to the User, under the disguise of being the generic ones for any equivalent user, contain specific obscure conditions that the User is unaware of. The Service thus cheats by crafting a special Agreement.

– *Service shows correct Terms but invisibly modifies immediately after technically obtaining consent.* This involves how consent is recorded such as in a webpage form. It is trivial to display the user a document that is then invisibly manipulated without the User being aware of.

– *Dishonest user claims consent to different Terms.* We assume it is ultimately up to the Service to make sure the User is fully aware of all Terms which includes the user interface; for example, the users device must be able to display the page properly and this check should be done programmatically as much as possible. The User can always argue that she or he never accepted some Terms or that they were visually manipulated or hidden.

The following are outside the scope of this paper:

– *Modification of in-flight packets.* We simply assume that all parties are communicating over secure channels such as HTTP/TLS.

– *Impostor using someone elses Data.* As said, we cannot verify the user at any level and as such we can only define the User as the person engaging the Service behind a particular device at a given time.

– *Impairment of Users device.* We also leave out of scope the possibility of the users device, deliberately or not, not having the expected functionality. For example, a component (such as a browser add-on) might be interfering in the way the Agreement is executed.

4 Strategies

We propose two strategies that differ when considering the trade-off between usability and hardness of guarantees.

– *peer-to-peer (p2p).* We have a simpler 2-party, peer-to-peer scenario, between the User and the Service collecting personal information. No other party is involved. As discussed, there is no known and fair way to obtain hard assurances between two parties that need to exchange something. In simple words, someone has to send first and, as discussed, progressive schemes (where parties exchange shares incrementally) only provide some level of assurance. In the p2p case, we assume the User is not particularly concerned if, after sending the personal data, the service does not send a receipt.

– *Trusted Third-Party.* If the User is concerned about sending personal data without guarantees, one can use a Trusted Third Party (TTP). In essence, the personal data can be first sent to the TTP, for which an evidence of submission is issued. The Service will then send evidence of *commitment to receive* the personal data on which the personal data is released. Several types of TTPs exist but we assume the risk of it being a bottleneck is not acceptable and therefore an offline TTP (who is only engaged if a party cheats) is used.

4.1 Peer-to-Peer

The peer-to-peer case assumes that a user U accepts the risk that service S will not return the expected service and only exists to collect personal data disappearing immediately after. This relaxation simplifies the problem but is not strictly a fair exchange.

User U starts by gathering and locally storing Agreement $\mathbf{A}_u = \{D_u, L, T\}$ where D_u is the user's personal data, L is a representation of the collection logic and T is the Privacy Policy or Terms of Agreement. U then applies a one-way, collision resistant, function to produce $H_u = hash(\mathbf{A}_u)$. U then generates a pair of public/private keys, K_u^{pub} and K_u^{priv}. This pair is, effectively, what binds the physical person to the consent.

User U then sends message $M_u = \{\mathbf{A}_u, Enc_{K_u^{priv}}(H_u, t_u, r_u), t_u, H_u, r_u, K_u^{pub}\}$. This message consists of the agreement $\mathbf{A}_u$, its hash H_u, a timestamp t_u and a sufficiently long random nonce r_u. The message is further encrypted with K_u^{priv} and is sent along with the corresponding public key K_u^{pub} so S can verify. The role of r_u is that, along with another that S will send, r_s, provides a (with high likelihood) unique identifier of the particular context of this agreement. Further note that, from the perspective of S, the user is completely identified as the person who owns K_u^{priv}. In this sense, encryption is here used to create a notion of identity and provide authenticity in the short-lived session during which this agreement is negotiated.

Service S runs a similar procedure. It starts by gathering $\mathbf{A}_s$ independently and then calculating a hash $H_s = hash(\mathbf{A}_s)$. It then verifies H_u matches H_s. S is now assured U returned the offered agreement. S further generates K_s^{pub} and K_s^{priv} as, respectively, a pair of public and private keys. In a common scenario, these keys could be those associated with a PKI certificate which fairly resolves the problem of S's identity. For example, if the negotiation is over HTTPS, the keys associated with the TLS certificate could be used.

Similarly, S sends $M_s = \{\mathbf{A}_s, Enc_{K_s^{priv}}(H_s, t_s, r_s), t_s, r_s, K_s^{pub}\}$. U becomes assured S signed the expected agreement. U verifies H_s matches its own H_u which assures U that S returned the expected agreement. At this stage, we define $H = H_s = H_u$ as both parties verified.

U sends evidence $E_u = \{Enc_{K_s^{pub}}(H, r_u, r_s, t_u, t_s)\}$ and attaches its signature $S_u = Sign_{K_u^{priv}}(E_u)$. Together with the agreement gathered at the first step, S generates $\mathbf{R}_s^t = \{\mathbf{A_s}, E_u, S_u\}$ which is the receipt that S must hold as proof of consent.

Symmetrically, S sends evidence $E_s = \{Enc_{K_u^{pub}}(H, r_u, r_s, t_u, t_s)\}$ and attaches its signature $S_s = Sign_{K_s^{priv}}(E_s)$. Together with the agreement gathered at the first step, U generates $\mathbf{R}_u^t = \{\mathbf{A_u}, E_s, S_s\}$ which is the receipt that U must hold as proof of consent.

4.2 Using an Optimistic TTP

Using a TTP TTP has the advantage that user U does not have to send personal data before obtaining confirmation of receipt and risk a cheating service S that does not continue the expected protocol. It is also easy to design a scheme where TTP also does not have access to the personal data and serves merely as a trust binding channel that is called upon only if any party does not complete the protocol – hence called *o*ptimistic. It is still possible for a cheating S to obtain the personal data and not deliver the expected service but U will have evidence of it, contrary to the pure peer-to-peer. Our scheme is inspired in [13].

U runs the first steps of the previous strategy but instead of sending the agreement $\mathbf{A}$ in plaintext, it is sent encrypted with a symmetric key chosen for the session k. This key is also sent but encrypted with TTP's public key, K_{TTP}^{pub}. U sends therefore $M_u = \{Enc_k(\mathbf{A}_u), Enc_{K_u^{priv}}(H_u, t_u, r_u), t_u, H_u, r_u, K_u^{pub}, Enc_{K_{TTP}^{pub}}(k)\}$. Note that the TTP has no access to the personal data.

If S responds with a receipt, U sends k which S should then acknowledge with a final receipt. In case S does not send the final acknowledgment, U submits (k, r) to TTP which then sends to both U and S a receipt of submission.

4.3 Analysis

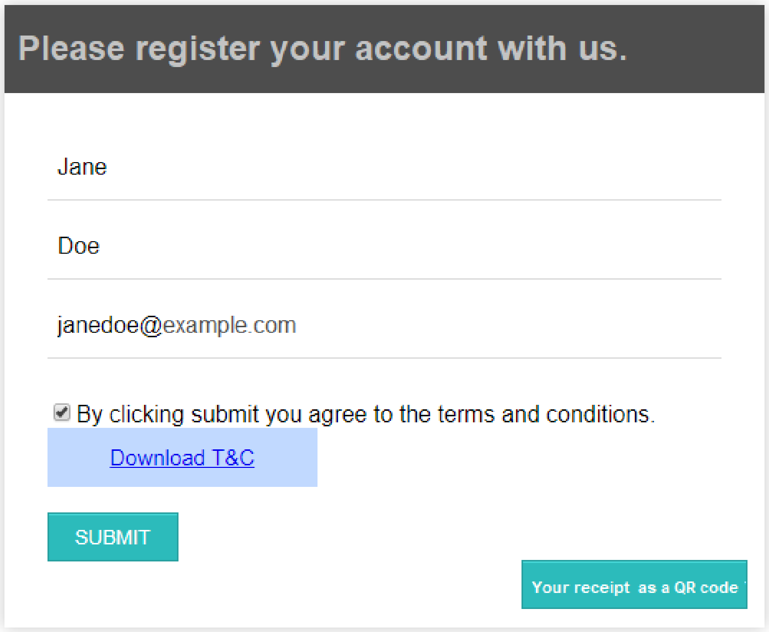
Assume that at a point in the future, either U or S challenge the agreement in place, by fully or partly denying it.

In the p2p case, the scenario is symmetric so we will look only at U denying ever accepting, in whole or in part, the agreement. S can trivially prove, or force U to withdraw the challenge, by requesting the evidence on U's side. On producing evidence, with a signature based on S's public certificate, the nonce r, timestamps and identity of the service will uniquely identify the session. U can produce the agreement that was signed on its side and, along with signatures and hashes, uniquely identify both the collection logic, the personal data sent and the terms. Either S produces a consistent agreement or refuse to produce evidence at which point S has to be considered at fault.

In the TTP case, U will have evidence of submission since it shared the symmetric key k demonstrating it followed the protocol yet S did not. U has evidence of early submission so S either admits breaking the protocol or must produce its own evidence at which point the TTP will be able to open with its private key.

5 Evaluation

We implemented the peer-to-peer protocol in order to understand its impact on usability. The User was presented with a simple form as shown in Figure 2. Upon submitting the personal data and running the protocol, a button for a Quick Response (QR) code is to be available in order to both download a file containing all the components of the agreement and the cryptographic material (evidence of agreement) as detailed in section .


Please register your account with us.

Jane

Doe

janedoe@example.com

☑ By clicking submit you agree to the terms and conditions.

Fig. 2. User interface.

We used a common mid-spec laptop to run the protocol: Windows 10 Pro, Intel Core i5, 8Gb RAM. The browser running the javascript implementing the protocol on the client side was Google Chrome 72. We used open-source cryptographic libraries for javascript: Cryptico, Node-forge, js-sha256 and js-md5.

We tested different sizes of both the HTML and documents involve such as a Privacy Policy. The results we obtained are showing in Figure 5. We used two common hashing functions, SHA-256 and the older MD5. Results show that the impact in usability are minimal and fairly independent of the size of the agreement.

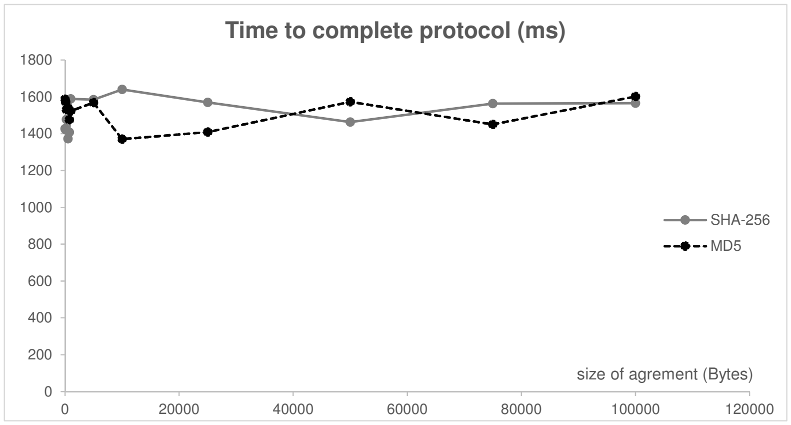


Fig. 3. Latency added by the fair exchange protocol.

All values add about 1.5 second more of latency which is on the order of magnitude of network latency for web pages. The size of the agreement does not seem to play a major role, at least in the range we considered (up to 100 kB of files). Also note that we cannot comment on how optimised the open source javascript libraries we used are. Finally, note that the same laptop was also running the web server and the Service protocol logic (in Node.js) so the network latency was minimal and well under 1ms. In a realistic setup, we expect the network latency (commonly on the 100ms) to dominate. For a one-off registration, these results suggest that a protocol of the kind we propose will not have a visible impact.

6 Conclusions and Outlook

In this paper, we tackled the challenging problem of obtaining poof of consent when sharing personal data. To the best of our knowledge, this is the first paper to do so. We show that, beyond the techno-legal aspects, Fair Exchange Protocols can be an invaluable aid towards this goal. We also showed that our proposal is feasible in terms of usability , despite improvements needed.

A number of open questions are raised. On one hand, personal data is not shared only on account creation but virtually at any stage, which is currently an intensely debated topic. Enabling proof of submission for real-time data may use the same approach as we do here (in a rather static scenario) but raises usability and manageability challenges - for example, an ultimately, each data packet could be the subject of a receipt which would dramatically impact the performance of the current Internet. Another direction we will be pursuing is how to completely encapsulate an Agreement when the collection logic is not open as it happens with mobile apps. Finally, our scenario, despite overwhelmingly used, considers only web applications. Work should be done at a larger scale with other types of applications, from mobile applications to machine-to-machine.

7 Acknowledgments

We would like to thank Professor Peter Sommer, of Birmingham City University, for his insights into the legal aspects.

References

1. Millett LI, Friedmann B and Felten E., Cookies and web browser design: Toward realizing informed consent online. In Proc of the Conference on Human Factors in Computing Systems 2001
2. Tuomas W. Sandholm, Unenforced E-Commerce Transactions, IEEE Internet Computing, vol. 1, no. 6, November 1997
3. Thilla Rajaretnam, The problem to consent to the collection, use, and disclosure of personal information in cyberspace. In International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012
4. U. Maurer, New approaches to digital evidence, in Proceedings of the IEEE, vol. 92, no. 6, pp. 933-947, June 2004.
5. Ben Laurie and Nicholas Bohm, Signatures: An interface between law and technology, January 2003. Online: http://www.apache-ssl.org/tech-legal.pdf
6. Jianying Zhou and Dieter Gollmann, A fair non-repudiation protocol. In Proceedings of the 1996 IEEE conference on Security and privacy (SP'96), Washington, DC, USA, 1996
7. Jaroslaw Watrobski, Artur Karczmarczyk, Application of the Fair Secret Exchange Protocols in the Distribution of Electronic Invoices, Procedia Computer Science, vol 112, 2017
8. Alois Paulin and Tatjana Welzer, A universal system for fair non-repudiable certified e-mail without a trusted third party, Computer Security, no.32, February, 2013
9. W. Neville and M. Horie, Efficiently Achieving Full Three-Way Non-repudiation in Consumer-Level eCommerce and M-Commerce Transactions, IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications, Changsha, 2011
10. B. Garbinato and I. Rickebusch, Secure multiparty computation vs. fair exchange: Bridging the gap, Technical Report DOP-20070123, University of Lausanne, DOP Lab, 2007. Online: http://www.hec.unil.ch/dop/Download/articles/DOP-20070123.pdf.
11. J. Onieva, J. Zhou, and J. Lopez, Multi-party non-repudiation: A survey, ACM Computing Surveys, vol 41, no. 1, December 2008

12. McDonald A and Cranor L F., The Cost of Reading Privacy Policies, A Journal of Law and Policy for the Information Society 2008; Privacy Year in Review issue I/S
13. O. Markowitch, S. Kremer, An Optimistic Non-repudiation Protocol with Transparent Trusted Third Party, In: G.I. Davida, Y. Frankel (eds), Information Security, ISC 2001, Lecture Notes in Computer Science, vol 2200. Springer
14. Jianying Zhou, Robert H. Deng, and Feng Bao, Evolution of Fair Non-repudiation with TTP, In Proceedings of the 4th Australasian Conference on Information Security and Privacy (ACISP '99), London, UK, 1999.
15. B. Garbinato and I. Rickebusch, Impossibility results on fair exchange, In Proceedings of the 6th International Workshop on Innovative Internet Community Systems (I2CS06), volume LNI. German Societyof Informatics, 2006.